Analytics Report PSR

ISSUE 3 JUNE 2026

Artificial intelligence is rapidly becoming a core capability in energy analytics, not only as a tool for automation, but as a new layer for modeling, reasoning, software development, and large-scale computation. This issue of the Analytics Report highlights how PSR is positioning itself at the forefront of AI applied to energy systems, combining climate scenario generation, stochastic optimization, reinforcement learning, agentic workflows, MCP-based interaction with analytical tools, AI-assisted coding, and GPU-based optimization. Together, the articles show how AI and advanced computing can support more robust methodologies, faster studies, and new ways of developing, operating, and interpreting complex energy models.

Special Issue

PSR User Meeting 2026

Full coverage of the PSR User Meeting 2026, held in Foz do Iguaçu, Brazil, which gathered participants from four continents around the event's two central themes: artificial intelligence in energy analytics and the increasing complexity of power systems. Highlights include the evolution of the SDDP Platform, client cases from around the world, and a memorable technical visit to the Itaipu hydroelectric plant — plus an invitation to the 2027 edition.

inBrief

AI-enabled SDDP Platform experience

A practical look at how chatbots, retrieval-augmented generation, MCP-based agents, and reasoning workflows can extend the SDDP Platform experience, supporting users from documentation queries and case operations to simulation execution and result analysis.

inSight

AI, climate modeling and software engineering

A practical look at how chatbots, retrieval-augmented generation, MCP-based agents, and reasoning workflows can extend the SDDP Platform experience, supporting users from documentation queries and case operations to simulation execution and result analysis.

inDepth

Advanced AI and computing for optimization

Technical articles on hybrid optimization and reinforcement learning for hydrothermal dispatch, AI-assisted implementation of stochastic generation-expansion solvers, agentic reasoning applied to hydrothermal operation, and GPU-based algorithms for large-scale optimization.

Editorial

This issue of the Analytics Report focuses on AI methodologies and applications to energy systems; to increased productivity for studies; and to the AI-assisted development of analytical tools ("math and coding").

The articles follow roughly the timeline ("milestones") of AI breakthroughs. The first milestone was Deep Neural Networks (DNNs). This will be illustrated with the generation of integrated scenarios of inflows, renewable generation and temperature (which affects load, equipment performance and multiple uses of water such as irrigation). These scenarios are produced by layered DNNs trained on Global Circulation Models (GCMs), which allows them to incorporate the impact of climate change on their probability distributions. These AI-based scenarios are used as input to PSR's analytical tools such as SDDP using joint autoregressive (AR) and Markov chain modeling that captures their complex nonlinear temporal and spatial relations.

The second milestone was Deep Reinforcement Learning (DRL). We show that there is a strong methodological relationship between DRL and the SDDP algorithm. This relationship was leveraged in the development of SDDeeP, a new analytical tool for operations planning that incorporates the "best of two worlds".

The third milestone was the development of the so-called Generative, or Large Language Models (LLMs) illustrated by ChatGPT. We show how LLMs are used to develop a set of tools such as comprehensive AI-assisted user support and AI-enabled extensions of the SDDP Platform experience, combining chatbots, MCP-based agents and reasoning workflows to support users from model documentation and case operations to simulation execution and result analysis.

The fourth milestone is related to the so-called "reasoning" models such as Claude and Gemini. In a very simplified way, they result from the combination of the two previous milestones, DRL and LLMs. We show how these reasoning models are used to accelerate both the development of new analytical tools ("math") and their implementation ("coding").

The fifth - and very recent - AI development is the so-called "agentic AI", where several autonomous models ("agents") use "skills" to solve a wide range of problems. We illustrate the application of agentic AI with a case study where the agents "discover" by themselves how to optimize the stochastic operation of a hydrothermal system with no knowledge about stochastic optimization methods.

The final topic in this report is the use of GPU-based architecture to solve large-scale optimization problems which are currently solved by CPUs. NVIDIA and PSR have an ongoing partnership where our optimization tools are run on NVIDIA's computational resources with support from their technical team. The first joint PSR- NVIDIA papers on the results will be released soon.

We hope you enjoy this issue.

The PSR Analytics Team

Special Issue

PSR USER MEETING 2026

Luiz Carlos da Costa Junior

From May 24th to 29th, PSR hosted the 2026 edition of its Global User Meeting in Foz do Iguaçu, Brazil — a setting that could hardly be more symbolic for a community dedicated to energy analytics, at the meeting point of three countries and next to one of the world's largest hydroelectric plants. The event gathered 90 participants from 16 countries and 46 organizations, joined by 54 specialists from PSR's own team. With representatives from Latin America, North America, Europe and Oceania — including system operators, regulators, planning agencies, utilities, market participants, consultancies, universities and development institutions — the meeting consolidated its role as an international forum for technical exchange, combining formal sessions with informal interaction to strengthen collaboration across the energy community.

AI and complexity: the threads running through the week

The 2026 program was built around real challenges faced by the institutions in the room, connecting those problems to methodologies, models and computational solutions that support decisions in increasingly complex environments. Two central themes ran through the entire week.

The first was artificial intelligence and analytical workflows, examined from both sides of the equation. As a new driver of demand, AI brings data centers and large flexible loads to the core of the planning agenda. As a source of opportunity, the program traced the evolution from neural networks and deep reinforcement learning to generative AI, reasoning models and agents, and showed what this means in

practice: agents interacting with planning models through MCPs and chatbots, GPU-accelerated solvers for large-scale optimization, AI-assisted demand forecasting, and a new generation of workflow automation that accelerates the development of analytical solutions and changes how users interact with models, data and results.

The second was the increasing complexity of power systems: renewable integration down to sub-hourly variability, security of supply and resilience in renewable-dominant systems, climate change in expansion planning, transmission modernization, hydrothermal operation, the value of coordinated operation, and market design transitions under decarbonization and geopolitical uncertainty — all connected directly to planning, operation and investment decisions being taken today.

Strategic opening: energy, geopolitics and intelligence

The opening plenary set the tone for the week with a broad strategic discussion on the forces reshaping energy systems worldwide. After welcome remarks by Raphael Chabar, PSR's Executive Director, the session brought together keynote perspectives from Rob West, founder of Thunder Said Energy; Pascal Van Hentenryck, head of Gurobi's AI Innovation Lab and Director of the US NSF AI Institute for Advances in Optimization at Georgia Tech; and Mario Veiga Pereira, PSR's founder and creator of the SDDP algorithm.

The discussion placed artificial intelligence at the center of the sector's strategic agenda — and on both sides of it. Rob West described what he called an "AI energy transition," in which solar, batteries, robotics and AI reinforce one another: data centers are driving electricity demand growth at a pace not seen in decades, but AI deployed across industrial assets and grid operations can unlock efficiency gains and additional capacity from existing infrastructure of comparable magnitude. Pascal Van Hentenryck and Mario Pereira connected this to the analytical challenge: power systems must now co-optimize generation, transmission, storage and demand under deepening uncertainty — from climate-driven hydrological volatility to geopolitical fragmentation — which raises the bar for optimization methods and decision-support tools alike.

A second block broadened the conversation to climate, planning and market design. It opened with a deliberately provocative video by Dave Borlace, creator of Just Have a Think, one of the world's most-watched independent channels on clean energy and climate — presenting recent research on accelerating warming and rising tipping-point risks as a trigger for debate. Mario Pereira and Philippe Dunsky, founder of Dunsky Energy + Climate Advisors and chair of Canada's Electricity Advisory Council, reacted with their views on what these climate signals mean for decision-making, planning and the analytical tools the sector relies on. The live discussion, moderated by Raphael Chabar, then had Ricardo Mota Palomino, former Director General of Mexico's CENACE, on stage, and Anders Kringstad, Director for Long-Term Market Analysis at Norway's Statnett, connected remotely — reacting to video perspectives from Luiz Augusto Barroso, PSR's CEO, on planning and institutional decision-making; Rodrigo Moreno, of the University of Chile and Imperial College London, on transmission expansion in Latin America; Carlos Batlle, of Comillas University and MIT, on electricity market design; and David Victor, of UC San Diego, on geopolitics and the energy transition. The result was a panorama that balanced strategic vision with methodological substance — and that framed the technical program of the days that followed.

SDDP 19: the new generation of the SDDP Platform

The session dedicated to PSR's analytical tools opened with a clear diagnosis of how much the modeling challenge has changed: renewables have multiplied variability and uncertainty in virtually every system — no longer only in hydro-dominated ones; transmission bottlenecks proliferate as generation moves to where the resources are; data centers and large flexible loads introduce demand dynamics never seen

before; and the operational detail required by studies has outgrown traditional tools and workflows. Against this backdrop, PSR presented SDDP 19 — the new generation of the SDDP Platform, an integrated environment connecting long- and short-term operation planning, expansion planning, transmission-aware analysis, data management, visualization and collaborative workflows.

A major highlight was NCP 7.0. PSR's short-term dispatch tool — used for decades in control centers of several countries for day-ahead and week-ahead scheduling with redispatch, and long integrated with SDDP — now becomes part of the SDDP Platform, detailing SDDP's mid- and long-term results down to operational granularity. With configurable resolution from sixty minutes to one minute, NCP adds a layer of operational detail beyond the hourly resolution — intra-hour solar ramps, battery arbitrage, reserve activation — with unit-level hydro representation, detailed thermal constraints and hybrid renewable-plus-storage plants, across three complementary use cases: day-ahead scheduling, intra-day redispatch and near-real-time balancing. Its full methodological consistency with SDDP — including water values and other medium- and long-term strategic decisions passed directly from SDDP's stochastic policy — connects the long-term policy to minute-level execution within a single analytical chain.

The session also introduced Foresight, PSR's new demand forecasting tool, covering three horizons — long-term for expansion planning, mid-term for price and market studies, and short-term hourly forecasting for operations — with systematically benchmarked statistical models, segmentation by consumption class, macroeconomic and climate drivers, and the ability to represent emerging loads, from electric vehicles to data centers, for which no historical series exist.

SDDP itself, the operation planning tool, received important methodological and computational advances: a penalty-free formulation based on feasibility cuts, which detects true infeasibilities while preserving consistent price signals; explicit inertia constraints, bringing frequency response into dispatch and investment decisions; refined battery representation with cycle counting, cycling limits and self-discharge; and a redesigned cut management engine delivering substantial performance gains on large-scale cases. OptGen, the expansion planning model, now explicitly represents data centers and large flexible loads, incorporates technology degradation to avoid biased investment choices, and automatically calculates the firm capacity of renewables — capturing how the capacity contribution of each technology changes as penetration grows and peak hours shift — alongside convergence improvements, network reduction and integration with PSRPlot dashboards.

Transmission received special attention, reflecting its new role as a central constraint on energy transition. Building on the platform's detailed network representation available since SDDP Platform 18 — AC elements, transformers, FACTS devices, DC links and dynamic line ratings —, the NetPlan algorithms and modules, applied for decades in transmission studies and in which PSR has deep, long-standing expertise, are now incorporated into the SDDP Platform: OptNet for least-cost transmission expansion, OptFlow for network-constrained optimal power flow, and OptVAr for reactive power and voltage support assessment. Energy Map will be the platform's new georeferenced visualization tool, conceived from the ground up for complex, large-scale networks, with single-line diagrams, layered result analysis and map-based editing.

The user experience also took a leap forward. The new Time Series input format accepts free-form CSV files with flexible structures, sparse and periodic dates and date patterns, automatically converting data between resolutions; PSRIO consolidates its role as a high-performance business intelligence tool for simulation outputs; PSRPlot brings a fully integrated charting and dashboard experience; the new PSR Knowledge Hub reorganizes documentation around products and learning paths, with improved search and video tutorials; and the SDDP Data Manager brings version control to SDDP cases — historical versions, change tracking and collaborative sharing — making studies more traceable and reproducible. Deep-dive demonstrations later in the week allowed participants to explore these capabilities directly with the teams that develop them.

Artificial intelligence takes center stage

If one theme defined this edition, it was artificial intelligence. A dedicated session — linking back to the strategic discussion that opened the week — presented AI as a new interaction and orchestration layer for advanced analytical models: not replacing mathematical optimization, simulation and expert judgment, but expanding how we formulate problems, create software, explore results and automate workflows.

The session was built as a progression. It began with a retrospective of how AI got to where it is — from its origins through neural networks and deep reinforcement learning to today's generative AI, reasoning models and agents — giving every participant the context for the discussions that followed, and connecting this trajectory to PSR's own research, including GPU-based algorithms for large-scale optimization developed in collaboration with NVIDIA. From there, the program moved to an experiment posed as a challenge: can an AI agent learn to operate a hydrothermal system? Given access to domain capabilities exposed through MCP, but no dedicated optimization algorithm, a reasoning agent explored the system, estimated water values and built its own operating policy — arriving remarkably close to the performance of dedicated stochastic optimization. The message was architectural rather than triumphal: the value lies in placing AI inside the analytical loop, orchestrating trusted models, not in replacing them.

The final block brought these paradigms to day-to-day practice. PSR presented the expansion of the SDDP Platform experience through AI — chatbots, retrieval-augmented documentation, MCPs and reasoning workflows supporting everything from data queries to simulation execution and result analysis — and a client showed its own MCP-based application connecting SDDP and NCP to support short-term operational decision-making, a sign that users are not waiting to build on these capabilities.

Planning for complexity: transmission, data centers, hydro and beyond

If artificial intelligence was the first thread of the week, the second — the increasing complexity of power systems — ran through a sequence of technical sessions in which PSR's teams showed how that complexity translates into methodology and studies.

Transmission had a dedicated session, moving from the platform's grid-aware capabilities to applied studies: an integrated workflow combining expansion planning with detailed electrical studies, illustrated by a binational interconnection assessment that carried stochastic dispatch scenarios all the way to AC power-flow validation; and a dynamic line rating study showing how explicitly modeling weather-dependent transmission capacity, within the same stochastic framework, can unlock significant additional capacity from existing lines and reduce supply risk — a concrete example of analytics extracting value from infrastructure that already exists.

Data centers and large flexible loads — the demand side of the AI story — received a session of their own, covering the Brazilian and international context, the technical and regulatory challenges of connecting these loads, and an integrated methodology using SDDP and OptGen to identify optimal connection points and expansion and operation strategies, including the representation of elastic demand, flexible demand and large demand blocks.

Hydro, a foundation of the community since its origins, returned with a forward-looking agenda centered on HERA, PSR's environment for planning hydroelectric reservoirs and pumped-storage plants. A session on customized solutions showed how PSR's analytical core extends beyond the power sector itself, with applications built for client-specific decisions in gas logistics, biofuels, green hydrogen and steel production.

Finally, the value of coordinated operation and integrated planning anchored the discussion on decision-making. PSR presented its System Operator Value methodology — applied with national system operators in Brazil and Mexico to quantify, through counterfactual scenarios, the value that coordinated operation creates for society —, a strategic initiative on integrated generation and transmission expansion planning for the Brazilian system, a climate-aware expansion study for Colombia, and an analysis of the impact of representing renewable uncertainty stochastically rather than deterministically — evidence that modeling choices change decisions.

Client cases and international experiences

As in every edition, the heart of the User Meeting was the series of case studies presented by PSR's clients. System operators, planning agencies, regulators, utilities, generation and trading companies, and consultancies — shared how PSR's tools support real planning, operation and investment decisions. Taken together, the presentations covered the full spectrum of challenges facing modern power systems: grid evolution and decarbonization studies in Canada and New England; the implementation of SDDP by the Norwegian TSO and applications in Nordic markets; planning for a system operating with nearly fully renewable generation in Costa Rica; regulator-reviewed generation and transmission expansion planning cycles in Honduras; a long-term recovery and expansion plan for the Venezuelan power system; portfolio risk analysis integrating hourly generation and price forecasts in Brazil; investment-intelligence workflows delivering capture-price analysis on demand; battery revenue modeling in New Zealand; operational flexibility and ramping in Bolivia; hydroelectric potential assessment and floating solar on shared river basins; an AI-enabled user application connecting MCP, SDDP and NCP for short-term operational decision-making in Guatemala; and three decades of experience applying SDDP in Central America's emerging markets. The diversity of contexts confirmed, once again, the adaptability of PSR's analytical framework across regulatory structures, system topologies and stages of the energy transition.

These presentations also resonated strongly in the event evaluations. Participants repeatedly pointed to the user and international cases as among the most valuable content of the week — in the words of one attendee, seeing the tools applied in real companies, generating value and opportunity, is the best possible demonstration of their worth. The exchange of experiences between countries and institutions was singled out as one of the event's defining strengths, with many participants leaving with ideas drawn directly from their peers' presentations. And the growing appetite was visible on both sides of the stage: the number of client presentations more than doubled compared to the previous edition, and when asked about future editions, participants asked for even more space for user presentations.




Technical visit to Itaipu: a landmark experience

Among the most memorable moments of the week was the technical visit to the Itaipu hydroelectric plant. Walking through one of the world's most important hydropower assets — a plant that many participants have represented in their models for years — gave the community a rare opportunity to connect analytical practice with physical reality, observing at first hand the scale, engineering and operational context behind the data. The visit was repeatedly mentioned by participants as a high point of the event, and a fitting complement to a program in which hydro reservoirs, pumped storage and floating solar were recurring technical themes — including a case study presented by Itaipu Binacional itself. The on-site program was completed by visits to the Three Borders Landmark and to the Iguazu Falls, which offered participants shared experiences outside the conference room and reinforced the spirit of community that distinguishes the User Meeting.

Direct dialogue: talk with experts, software stands and one-on-one meetings

Beyond the plenary sessions, the agenda was designed to maximize direct interaction. Daily “Talk with Experts” sessions organized thematic tables around SDDP, OptGen, NCP, NetPlan, HERA, OptFolio and user requests, where participants discussed modeling approaches, customizations and new ideas directly with PSR specialists. Software stands during coffee breaks offered continuous live demonstrations, and individual meetings between clients and PSR staff provided space for in-depth, personalized dialogue. More than a conference, the event once again functioned as a collaborative forum — where real-world problems are discussed with the developers who build the solutions, and where users directly influence the future roadmap of the tools.

The view from the audience

The feedback collected at the end of the event was strongly positive across the board. Participants' evaluations placed overall satisfaction and willingness to recommend the event at the very top of the scale, with expectations widely exceeded. The technical quality of the content and the level of the speakers were the most praised aspects, together with the organization, the simultaneous translation and the social and technical program. The artificial intelligence session stood out as the most valued content of the week, and the international client cases were celebrated as proof of the value the tools generate in practice. Perhaps the most telling result: most participants stated that they intend to apply something they learned — new tools, AI-enabled workflows or new modeling approaches — in their work within the next few months.

Roadmap and closing

The closing session presented PSR's development roadmap organized around three fronts: streamlined execution and workflow automation; new modeling features and methodological advances in SDDP, OptGen, NCP and Time Series Lab; and new platforms and strategic applications. A structured interaction with the audience collected priorities and suggestions directly from users — input that will shape development decisions in the coming cycles — before the delivery of certificates and the final remarks from PSR's leadership.

See you in 2027

The PSR User Meeting 2026 confirmed the strength of a global community that shares not only tools, but a common analytical culture — and a common ambition to plan and operate energy systems better in an increasingly complex world. Planning for the 2027 edition is already underway, and we warmly invite all clients, partners and friends of PSR to join us for what promises to be another week of technical exchange. We hope to see you there.

In the meantime, the conversation continues: the next issues of the Analytics Report will feature dedicated articles exploring, in depth, the main topics presented at the PSR User Meeting 2026 — from SDDP 19 and the new platform capabilities to artificial intelligence in energy modeling and the client experiences shared during the event. Stay tuned!

inSight

INTEGRATED AI-BASED CLIMATE MODELING AND STOCHASTIC OPTIMIZATION

Luiz Barroso, Julio Alberto Dias and Mario Veiga Pereira

This article is organized into two parts:

Part 1 — PSRCast in energy analytics

The first part is based on a recent presentation delivered at the DTU PES Summer School in Denmark. It provides a comprehensive overview of PSRCast, PSR's AI-based scenario generation tool, and its integration with analytical tools such as SDDP.

The links to the material presented by Luiz Barroso and the corresponding video are below:

Presentation: https://www.psr-inc.com/app/link/?t=d&f=2026-05LuizBarrosoDTUSummerSchool.pdf

Video: https://www.youtube.com/watch?v=p7CLXFiqxCU

Part 2 — Applications beyond energy planning

The second part, written by Julio Alberto Dias and Mario Veiga Pereira, is adapted from a recent edition of PSR's Energy Report (https://www.psr-inc.com/en/energy-report/). It describes additional applications of PSRCast to agricultural insurance, sanitation and other areas, and discusses AI-based modeling of extreme events, which requires methodological adaptations compared with standard PSRCast modeling. A recent podcast complements this discussion: https://www.youtube.com/watch?v=U-7HrFSMSIg.

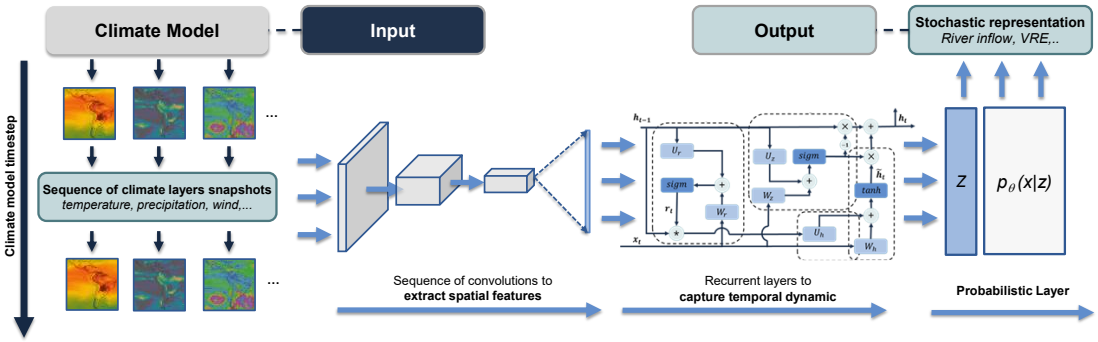
Among the various fronts of application of AI, climate modeling is today one of the central topics of greatest strategic interest. Whether improving forecasts, reducing biases in physical models, generating synthetic scenarios, or identifying complex patterns in large databases, machine learning techniques have significantly expanded the capacity to extract useful information from climate systems.

This movement gains additional relevance in the context of climate change. In a stationary environment, in which past behavior would be a good guide for the future, statistical models based exclusively on historical series could provide reasonable answers to various planning problems. However, the current climate system is in transition. Changes in atmospheric circulation, in the frequency of extremes, and in the ocean-atmosphere interaction make the hypothesis that the recent past adequately represents future conditions increasingly fragile. In this scenario, methods that simply reproduce historical patterns tend to become insufficient to capture new risk settings.

At the same time, the digital revolution has significantly expanded access to climate information. Projections from multiple global circulation models (GCMs), developed by different research centers, institutes and laboratories around the world, are now widely available. High-quality multi-model ensembles and reanalysis databases offer an unprecedented volume of climate system data. The challenge, therefore, stopped being the scarcity of information and became the ability to integrate it, interpret it and translate it into consistent and usable scenarios for decision-making.

Given the strong dependence of the electricity sector on weather conditions, PSR developed the PSRCast: an AI model designed with the objective of transforming projections of global circulation models into plausible scenarios of water resources, of the availability of variable renewable generation and of demand, consistent with the needs of operation and energy planning. It is structured to capture these relationships probabilistically, building a robust scenario generator, conditioned to global climate signals.

The following figure shows the architecture of the PSRCast.



As shown in the figure, the PSRCast is a Deep Neural Network (DNN) with three layers, where the input to the training set are the predictions of the GCMs and the outputs are the real events. This architecture allows the generation of thousands of climate scenarios in periods ranging from 15 days to ten years. PSRCast emulates the behavior of GCMs, but it can run thousands of times faster.

PSRCast applications

New climate-related energy planning and operation strategies for Brazil, Colombia, Mexico, Costa Rica, and other countries

Assessment of the climate impact on the reliability and prices of electricity supply (various customers)

Supply risk assessment for a water utility

New guidelines for climate risk reporting from utilities (partnership with BTG)

Climate risk assessment for agricultural insurance

Scenarios of extreme weather events of rain and wind gusts

As mentioned, PSRCast applications for energy were presented in an earlier edition of the Energy Report. Below we will show the last two applications on the list: agricultural insurance and extreme weather scenarios of temperature and wind.

Agricultural insurance

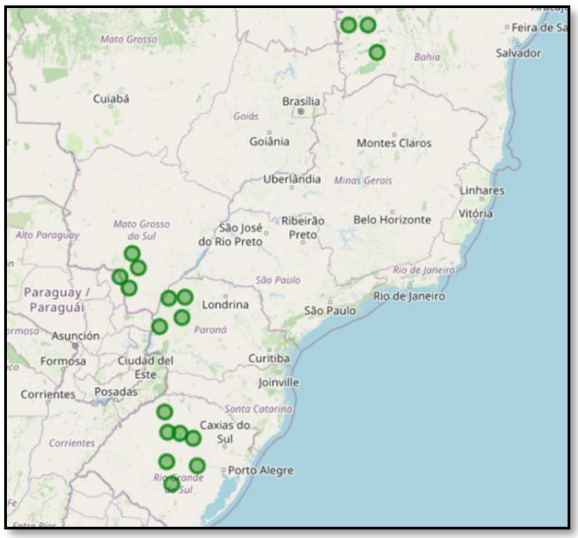
The PSRCast architecture goes beyond the specific application to the electricity sector; it can learn the relationships between GCMs projections and any quantity derived from these climatic signals. In this context, its performance can be understood as a probabilistic "downscaling" process, which translates the large-scale signals from global models to the scale of interest, adjusts structural biases in relation to the observed, and explicitly incorporates the uncertainty of the process.

One of the recent applications to which PSR has been contributing is the projection of agricultural crops, whose productivity over the cycles is strongly dependent on weather conditions. Variables such as maximum and minimum daily temperatures, accumulated solar radiation, wind and precipitation have a

direct influence on crop development and constitute fundamental parameters in agronomic simulation models.

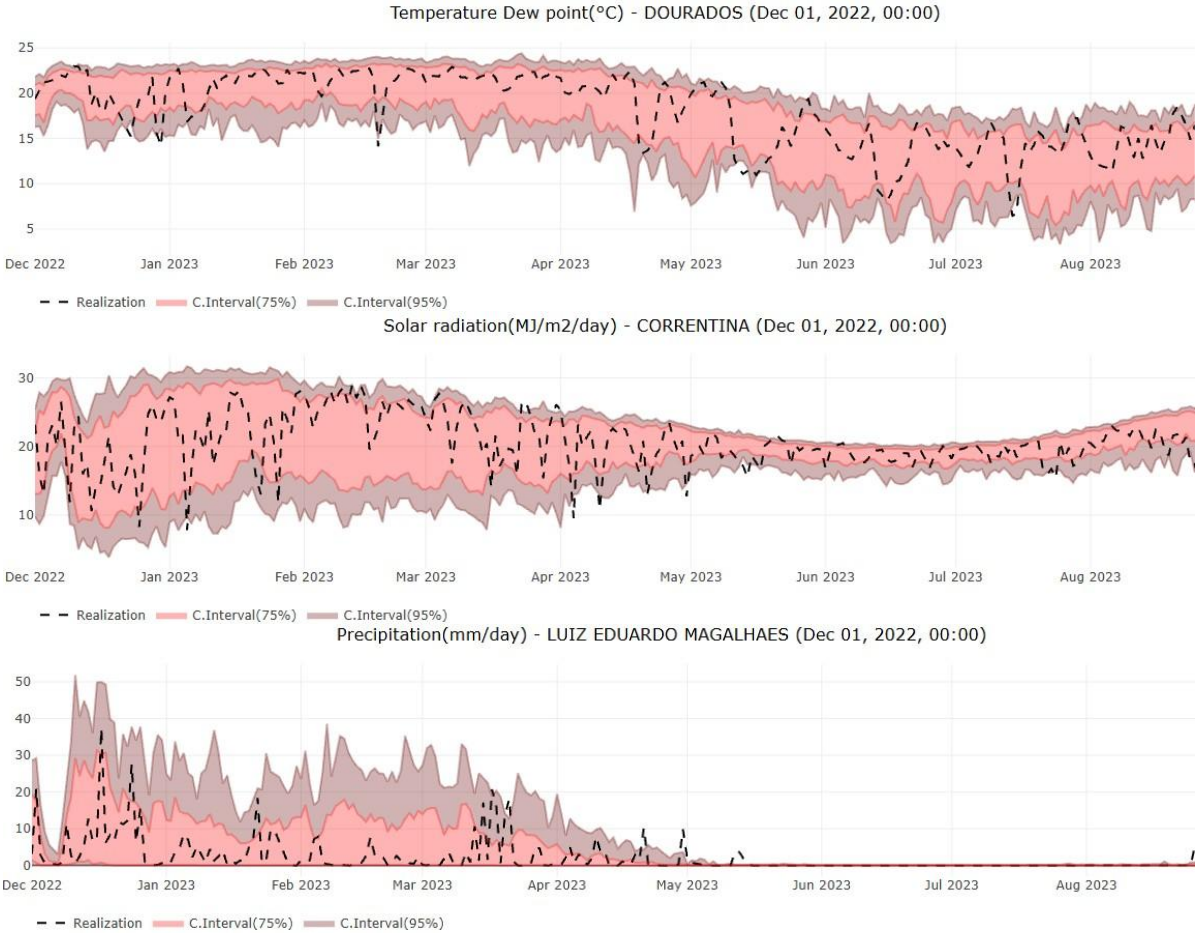
The following figure shows the geographical location of the regions analyzed in the context of this activity. The wide spatial distribution highlights the diversity of climatic dynamics involved, as well as the variety of agricultural crops considered, such as soybeans, corn and others, each with specific sensitivities to weather conditions.

In this application, the objective of the PSRCast is to generate hundreds of scenarios with daily resolution and a one-year horizon for a set of meteorological variables at all the points analyzed.



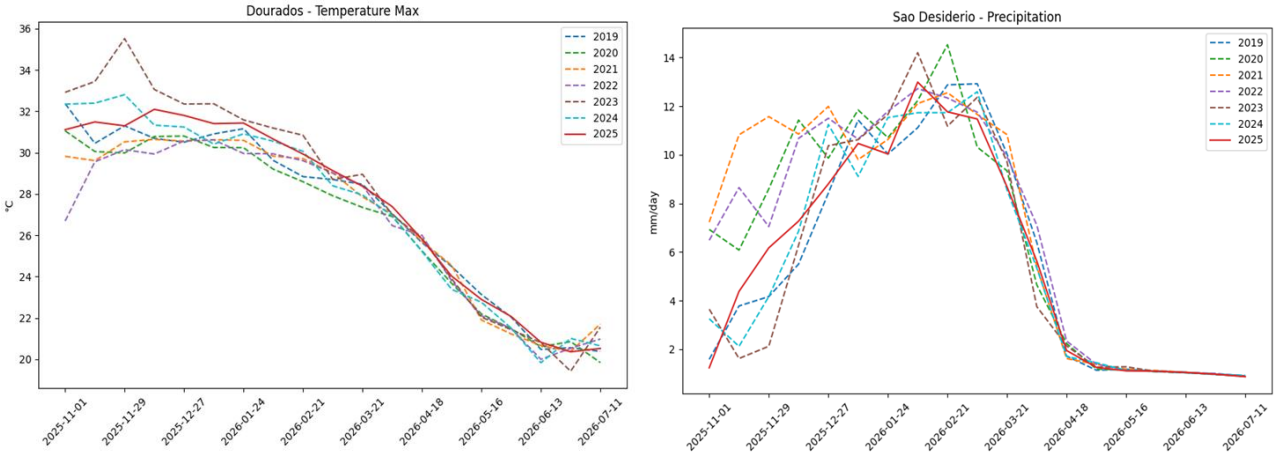
These scenarios are constructed in such a way as to preserve not only the individual statistical patterns of each variable, but also the spatial and temporal correlations between them, as well as other dependency structures that influence the development of crops throughout the production cycle. As mentioned before, the difference between PSRCast is that it does not use historical records to generate scenarios, because due to climate change, the past is no longer a representative reference for the future. And as readers can imagine, having the correct probability distribution of next year's events is critical to calculating agricultural insurance premiums.

In the following figure, examples of retrospective projections for the harvest cycle that began at the end of 2022 are presented, with the objective of illustrating, in a general way, the scenarios that would have been projected by the PSRCast compared to the values observed throughout the cycle.



It should be noted that, in the context of long-term projection, retrospective analyses not only serve to diagnose the adequacy of the model in the construction of consistent scenarios, but also to enable comparisons between the current projection and recent years. This approach allows a relative quantification of projected conditions compared to the recent past, providing an additional reference for interpreting risk and making decisions about the insurance premium.

This type of comparison is exemplified in the figure below.



These scenarios can then directly feed crop simulation models to estimate productivity. Because the scenarios generated preserve the natural statistical characteristics of the meteorological variables, as well as their spatial and temporal dependencies, they provide coherent and physically consistent inputs for these tools

Risk of extreme rainfall

Extreme rainfall events are among the main risk factors for the Brazilian electricity sector. Heavy and persistent rains directly impact dam safety, spill management, the integrity of civil structures, and the reliability of transmission and distribution systems. In scenarios of high climate variability, the ability to anticipate increases in the probability of extreme events becomes not only an analytical advantage, but a central element of risk management and the preservation of operational safety, especially in the current context in which we have already felt the impacts of climate change.

On the short-term horizon, that is, the next few hours or a few days ahead, the risk of extreme precipitation events is reasonably well anticipated by high-resolution regional weather models. These models explicitly resolve the dynamics of the atmosphere and its interaction with the surface with high spatial and temporal detail, making it possible to identify the formation and evolution of intense convective systems with good predictive capacity. At this scale, the combination of detailed physics and frequent assimilation of observational data provides a solid basis for weather alerts and immediate response actions.

The challenge shifts when the focus is on the quantification of risk over longer horizons, coming months, for example, in which strategic decisions must be taken under structural uncertainty. In this context, the analysis ceases to be a problem of deterministic prediction of a specific event and becomes an assessment of changes in the probability profile of extremes.

In a hypothetical scenario of unlimited computational capacity, it would be possible to probabilistically monitor the risk of extreme events by simulating a virtually infinite number of plausible atmospheric trajectories. Each trajectory would evolve according to the same physical equations used in meteorological models, allowing the probability distribution of extreme precipitation to be directly estimated based on the frequency of occurrence in the different simulated scenarios. In practice, however, computational limitations make it impossible to build ensembles of this magnitude, especially over a period of weeks to months. As an alternative, statistical approaches have been used to characterize the behavior of extremes, such as extreme value analyses (GEV models and Peaks-over-Threshold approach with Generalized Pareto distributions), adjustment of parametric distributions to the tails of historical series, in addition to methods based on resampling and stochastic generation of synthetic scenarios. While useful, these strategies rely heavily on structural hypotheses and tend to capture in a limited way the nonlinearity and multiscale dynamics that govern the occurrence of extreme events.

AI for extremes: same, but different.

At first glance, it may seem that structural monitoring of the risk of extreme precipitation events is just a natural extension of the AI model for generating long-term scenarios presented earlier. After all, if we can generate prospective scenarios for variables such as temperature, precipitation, wind, and flow, it would be sufficient to assess the distribution of these scenarios and extract the probabilities associated with the most severe events.

However, although there are methodological similarities, the nature of the problem imposes conceptually distinct challenges.

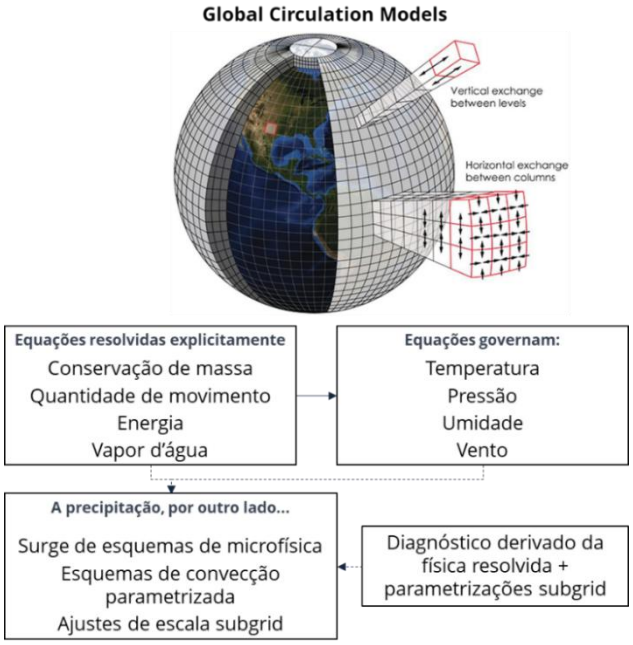
Briefly, two central peculiarities stand out. The first is the need to explicitly incorporate extreme event theory concepts into modeling. By definition, extreme events are rare, and their representation in historical data is limited both in frequency and in the diversity of atmospheric configurations. Models trained predominantly to reproduce average patterns or central distributions tend to underrepresent tails, precisely where the most critical risk resides.

The second peculiarity relates to the very nature of the rainfall projections over months' horizons. Ensembles derived from global circulation models are generally adequate to characterize average anomalies and expected seasonal patterns, functioning well as “drivers” for aggregated hydrological and energy projections. However, these ensembles were not designed to faithfully represent the physics associated with localized and intense extreme events.

In a previous edition of the ER, it was shown that the representation of extreme temperature and wind events with the support of AI can be understood intuitively (and simplified) by the following analogy: it is a process like the use of super-resolution models in images and videos. Just as AI algorithms learn to transform a low-resolution film into a sequence of sharper and more detailed “photographs”, preserving spatial and temporal coherence, models trained on climate model outputs can learn to refine temperature and wind grids, converting low-resolution atmospheric fields into more detailed representations. As these variables are states dynamically solved by the fundamental equations of the atmosphere, the “scaling” process tends to preserve coherent physical structures, allowing the most faithful reproduction of local extremes.

For example, this increased spatial “resolution” is being used to model wind gusts that can knock down transmission towers, in an ongoing R&D project.

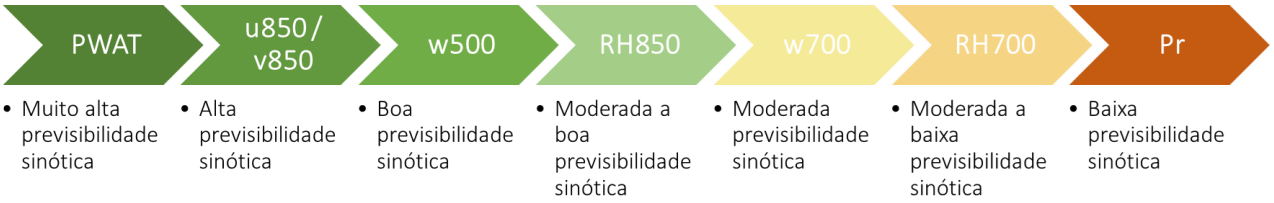
However, modeling extreme rainfall requires additional care. The reason is that, in GCMs, precipitation is not a variable solved directly by the primary physical equations, but a product derived from multiple parameterizations. This changes the nature of the problem, because the simple spatial refinement of the rain field does not guarantee the consistent reconstruction of the physical mechanisms associated with extreme events, requiring a conceptually different approach.



The long-term projection of precipitation is not the best explanatory variable for actual extreme precipitation.

Thus, instead of taking rain simulated by global models as a central explanatory variable, we can use an approach that shifts the focus to the atmospheric signals that precede and sustain the extremes. Monitoring is now conditioned by variables with greater synoptic predictability and greater structural stability, such as the total vapor content in the atmospheric column, the vertical movement at average levels of the troposphere, relative humidity at different pressure levels, and the winds not only on the surface, but also at specific atmospheric levels. Together, these fields represent the availability of moisture, dynamic vertical forcing, and organized steam transport, three fundamental physical pillars for the occurrence of extreme rainfall.

The following figure presents some of the main variables projected by global circulation models that help explain extreme precipitation events, as well as the degree of predictability of each one.



PWAT: Total water vapor content in the atmospheric column. Best indicator of maximum precipitation potential.

u850/v850: Zonal and southern components of the wind at 850 hPa. They represent organized moisture transport at low levels.

w500: Vertical air velocity at 500 hPa. Robust indicator of deep dynamic forcing associated with organized systems.

RH850: Relative humidity at 850 hPa. It controls moisture supply, but it is already under local influence;

w700: Vertical air velocity at 700 hPa. Useful for capturing convective intensification, but more sensitive to subgrid noise;

RH700: Relative humidity at 700 hPa. Additional information; greater variability and lower isolated robustness;

Pr: Precipitation forecast from the global model. Result of parameterizations; greater structural bias and lower robustness for specific extremes.

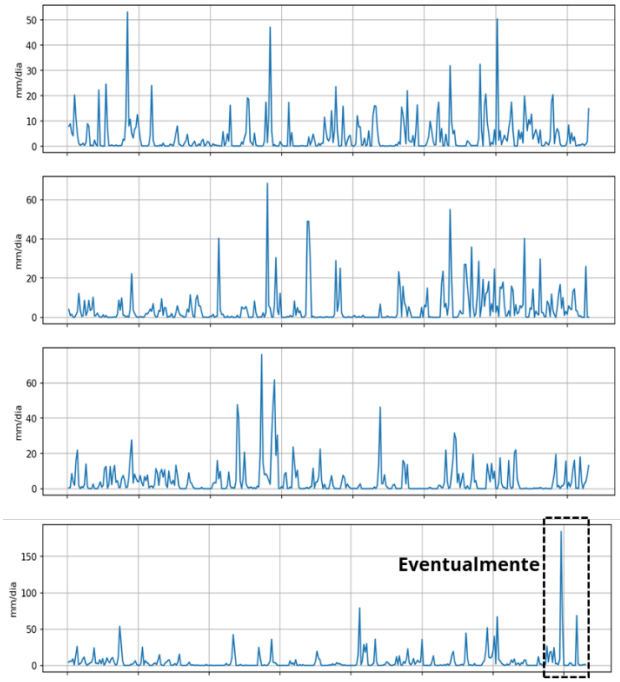
From this set of physically based conditionals, we can apply a Conditional Variational Autoencoders (CVAEs) architecture — the third layer of the PSRCast DNN, see previous figure — training them to generate consistent extreme precipitation scenarios at specific points of interest, preserving both physical coherence and observed statistical variability. With regard to the statistical representation, the structure of the CVAE must be adapted to more adequately reflect the asymmetric and intermittent nature of extreme precipitation. [1]

[1]Instead of assuming a traditional Gaussian parameterization, in which the latent space converges to a normal distribution, we propose a more sophisticated formulation, in which the generative process incorporates a Bernoulli—Gamma combination. In this configuration, the Bernoulli component explicitly

This strategy makes it possible to separate two levels of uncertainty: (i) the uncertainty associated with the large-scale atmospheric state, coming from global model ensembles; and (ii) the uncertainty associated with "subgrid" processes and local variability, captured by the probabilistic component of the AI model.

This approach is particularly consistent with the foundations of extremes theory, as it recognizes the highly asymmetric, heavy-tailed, and weather-dependent nature that characterizes intense precipitation events.

The figure below presents an illustrative real example in which the model was calibrated to generate consistent rainfall scenarios in a dam in the south of the country.



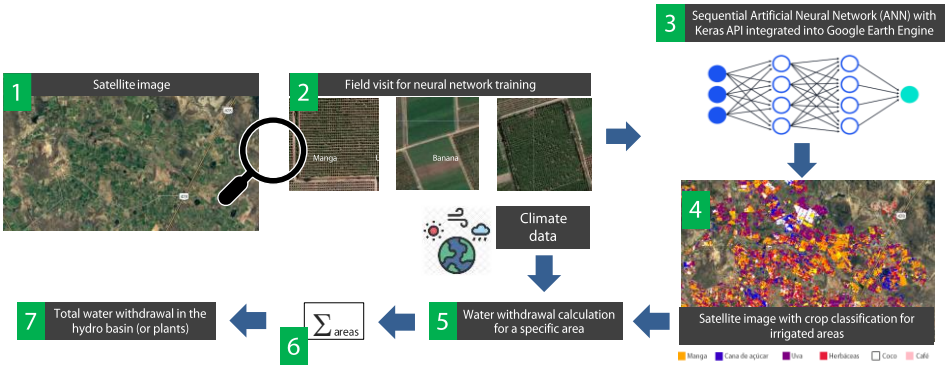
From different projected atmospheric conditions, it is possible to produce a massive set of daily rainfall sequences, allowing the construction of prospective risk distributions. These scenarios can be evaluated and compared to historical references and design parameters, making it possible to identify, for example, whether there are indications of an increase in the probability of occurrence of extreme events under current weather conditions or if the return times associated with critical rainfall considered when designing the dam remain adequately covered.

Use of DNNs and satellites to estimate water withdrawals for irrigation

The withdrawal of water for irrigation is very relevant information for the operation of the hydroelectric system. In a paper for Elera, PSR developed a methodology based on AI and information from electromagnetic spectra from the ground measured by satellites to accurately and automatically estimate this removal. The procedure is summarized in the following figure.

---

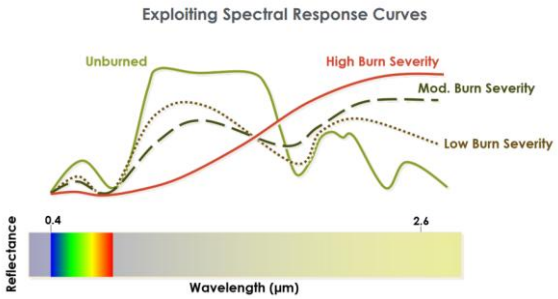
models the probability of the event occurring (rain versus no rain), while the Gamma component represents the distribution of intensities conditioned by the occurrence. Conditional atmospheric variables simultaneously modulate these two mechanisms: they alter both the probability of activation of the event and the parameters that govern its magnitude.

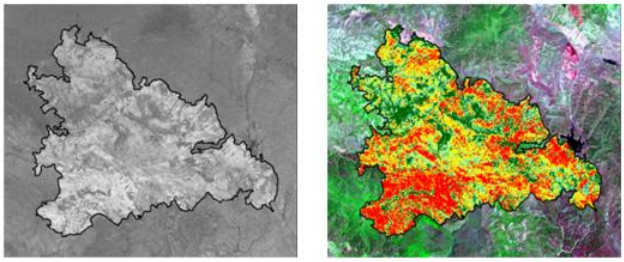


In a simplified way, spectral information makes it possible to identify each type of crop. This identification, in turn, makes it possible to calculate the water requirement. The final step is to estimate the need for irrigation by the difference between rainfall (produced by the PSRCast) and the need for water. The results proved to be quite accurate, and we are currently including the effect of evaporation from dams in the estimation and refining the measurements to shorter intervals. This information is of great importance for the better management of the so-called water energy food nexus, in the Northeast region.

Fire impact applications

As in the case of PSRCast, seen earlier, the same AI framework can be used for different applications. As the following figure shows, an analogous spectral analysis was used to establish the actual damage to the sugar cane crops that burned down in 2025. The model correctly identified the areas where the cane was "scorched" and where it was burned. This, in turn, allowed a better estimation of the impact on sugar and ethanol prices.





Conclusions

The applications of AI to climate modeling have allowed a significant gain in quality and resolution for the analysis of probability distributions of events affected by climate change. As shown, these applications extend beyond the energy area.

AI techniques have also been shown to allow the representation of extreme events such as floods and gusts of wind. Modeling these events requires additional care, but the results are equally significant in many areas.

The third area is the combination of AI with satellite information. The breadth of the results has been demonstrated again, and their relevance will only increase in the future with the increase in satellite measurements.

inDepth

SDDeeP: BLENDING OPTIMIZATION AND REINFORCEMENT LEARNING FOR HYDROTHERMAL DISPATCH

Gabriel Vidigal

Introduction

Hydropower is the world's largest source of low-carbon electricity, supplying close to one-seventh of global generation. Beyond its sheer scale, it is also one of the most flexible resources in modern power systems: large reservoirs can store water for months and release it on demand, smoothing seasonal variability and absorbing shocks elsewhere in the system. That same storage capacity, however, is what makes hydrothermal scheduling such a difficult planning problem. A decision to turbine or hold water today may not show its true cost or value until many months later, and that long memory must be reconciled with day-to-day operational realities such as transmission limits, thermal plant availability, and growing renewable variability.

For more than three decades, a central tool for this problem has been Stochastic Dual Dynamic Programming (SDDP), an algorithm deployed in operation planning studies in countries such as Brazil, Bolivia, Norway, Vietnam and the United States. In its classical form, SDDP relies on two structural assumptions: uncertainty must be represented in a way that preserves stagewise independence after a suitable transformation, and the underlying optimization problem must be convex. These assumptions are part of what makes the method tractable at scale, but they also shape the modeling choices available in practice, for example in how inflow processes are represented.

This article summarizes a recent study that explores an alternative path: combining optimization with Reinforcement Learning (RL) in a way that keeps the rigor of constraint enforcement while loosening the stagewise-independence assumption that SDDP imposes on the inflow model. The goal is not to replace SDDP, which remains a strong benchmark, but to understand what becomes possible when that structural restriction is relaxed, and at what cost.

Deep Reinforcement Learning in the Power Systems setting

Deep Reinforcement Learning (DRL) has gone through an unusually rapid maturation in the last decade. The field broke through with agents that learned to play Atari games from raw pixels, then went further with AlphaGo and AlphaZero, which mastered Go, chess and shogi entirely from self-play. More recently, the same family of techniques has become a routine ingredient in training large language models through reinforcement learning from human feedback. Each of these milestones pushed the algorithms forward: sample-efficient actor-critic methods such as Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3) and Soft Actor-Critic (SAC), and model-based variants that plan over an internal model of the environment. The result is a toolbox that is far more practical, far more stable, and far less data-hungry than what was available even five years ago.

Underneath the algorithmic variations, all RL methods share a common structure: a loop between an agent and an environment. At each time step, the agent observes the state of the environment and chooses an action; the environment transitions to a new state and returns a reward that scores the choice. Repeating this loop over the full horizon defines an episode. The agent's goal is to learn a policy (a rule that maps observed states to actions) that maximizes the cumulative reward over an episode.

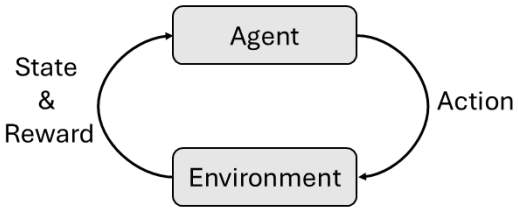


Figure 1 – The reinforcement learning framework

From a power-systems standpoint, what makes DRL interesting is its underlying kinship with the algorithms operators already use. Value-based DRL and SDDP are surprisingly close cousins: both alternate between a forward simulation step and a backward update that improves an estimate of the long-term cost-to-go function. SDDP represents that function with piecewise-linear convex cuts, which is what gives it its remarkable efficiency on convex problems. DRL replaces those cuts with neural networks, which are universal function approximators. Interestingly, when the network uses ReLU activations, as most modern architectures do, the resulting cost-to-go approximation is also piecewise linear, but no longer required to be convex. In other words, DRL can be read as a generalization of the same idea behind SDDP.

That generalization is starting to translate into concrete power-system applications. DRL has been used for real-time energy dispatch in IoT-driven microgrids (Lei et al., 2021), and recent surveys document a fast-growing range of additional use cases, including voltage and reactive power management on distribution feeders, bidding strategies in electricity markets, and the operation of battery storage in systems with high renewable penetration (Sivamayil et al., 2023).

The hydrothermal dispatch problem is another use case, marked by a multi-year horizon driven by reservoir time coupling, and by hard physical constraints on the network and on water balances. DRL's cost-to-go approximation can represent the nonlinear, time-coupled value of stored water across that horizon, but the constraints are a weak point for pure RL approaches. Pairing DRL with an optimization layer that enforces those constraints at each stage is what makes this hybrid approach viable in practice.

The proposed approach: target tracking with a learned policy

The method has two components that work in tandem. At each monthly stage, an optimization problem decides which thermal and hydro plants to dispatch, satisfying load balance, water balance, generation limits, and the full set of network constraints. What changes is how the long-term value of stored water is communicated to that single-stage problem. Instead of the piecewise-linear cost-to-go function used by SDDP, an RL agent (implemented here as a DDPG actor-critic) outputs a target reservoir volume for each hydro plant, together with a penalty weight that controls how strongly the operation should be pushed toward that target.

The single-stage problem then minimizes immediate operating cost plus a penalty on the deviation between the final reservoir volumes and the targets supplied by the agent. If the agent recommends keeping reservoirs full, water is stored aggressively; if the agent prefers to draw down, the optimization releases water to the extent the constraints allow. The simulated operating cost is fed back as a reward signal, and the actor and critic networks are gradually trained to recommend targets that minimize total cost across the full planning horizon.

Two design choices are worth highlighting. First, the optimization layer is what guarantees feasibility: the RL agent never has to learn what a network constraint is, because the optimizer enforces it for every dispatch decision. Second, splitting the problem in this way (single-stage optimization for the immediate decision, RL for the long-term coupling) removes the structural restrictions that multistage solvers

normally impose on the modeling. In particular, the inflow process can be any model that drives the simulation: historical data, a flexible time-series model, or any other process we believe better represents reality.

Figure 2 maps these pieces onto a single stage. The agent's input is the current state st (reservoir levels and recent inflow lags) together with the stage's realized uncertainties ωt (inflows and demand). Its action has two parts: a vector of target reservoir volumes ŝt+1 and a penalty weight βt that controls how strongly the optimizer should track those targets. Solving the state-tracking problem with these inputs yields the realized next state st+1 and the immediate operating cost ct, which is fed back as the reward signal used to train the agent.

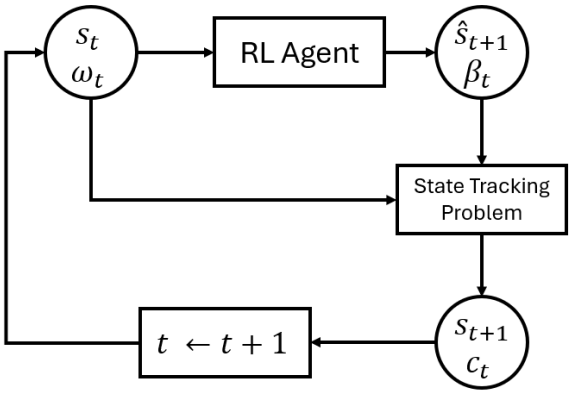


Figure 2 – Information flow between the RL agent and the single-stage state-tracking optimization problem.

Case study: inflow model

The proposed approach is benchmarked against SDDP in a setting where the structural assumptions of SDDP force a modeling simplification that the hybrid method can avoid: the stagewise-independent inflow model required by SDDP. The test system is the Bolivian power system (28 buses, 11 hydro plants, 23 thermal plants and 31 transmission branches), simulated over a 5-year horizon split into 60 monthly stages. Policies are compared on the same 10,000 out-of-sample inflow scenarios drawn from a SARIMA model fitted to the inflow history, and each RL configuration is trained five times with different random seeds to account for variability from neural network initialization.

SDDP was trained on scenarios from a PAR(p) model, as required by its stagewise-independence assumption. Three RL variants were trained on different inflow inputs: the artificial historical record itself, a SARIMA-based generator matching the out-of-sample data-generating process, and the same PAR(p) model used by SDDP.

The results show a clear ordering among the RL variants. Training on scenarios drawn from the true data-generating process, even a limited number of them, produced lower operating costs than training on PAR(p) scenarios, with a roughly 5% reduction. Drawing an unlimited number of SARIMA scenarios pushed costs down further, confirming that flexible inflow models combined with abundant training data lead to better policies. SDDP nonetheless finished ahead of all RL variants, with the best RL configuration landing within about 5% of the SDDP cost.

Two practical observations emerged. First, the inflow modeling assumption commonly used in operational planning is not innocuous: replacing PAR(p) with a richer stochastic process changed total operating costs by several percentage points, with every other assumption held fixed. Second, seed variability had a meaningful impact: different initializations led to different end-of-horizon reservoir levels, reinforcing the importance of evaluating multiple seeds in any practical deployment.

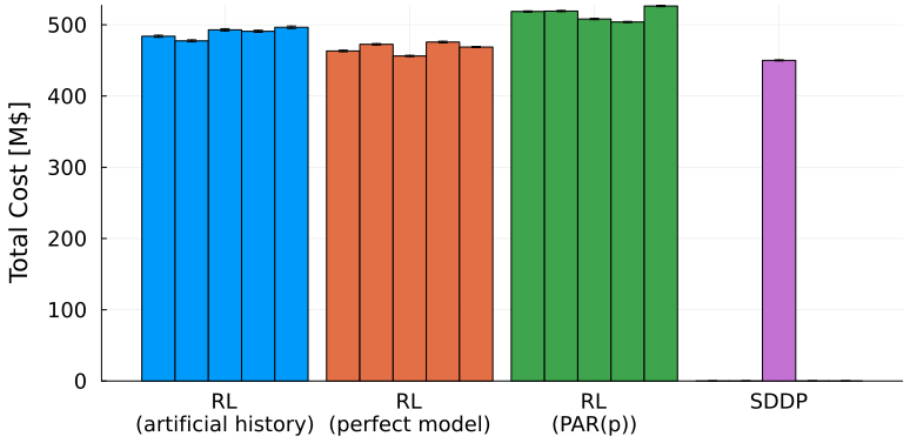


Figure 3 – Total operating cost across random seeds.

Discussion and outlook

The experiments support a measured conclusion. Combining optimization and Reinforcement Learning is a practical way to build hydrothermal dispatch policies without imposing the stagewise-independence assumption that SDDP requires on its inflow model. The single-stage optimizer keeps every physical constraint feasible, while the learned policy provides the temporal coupling that SDDP would otherwise enforce through its piecewise-linear cost-to-go function. Within that framework, training the policy on scenarios closer to the true data-generating process lowered costs in a consistent way across seeds, suggesting that the PAR(p) restriction routinely embedded in operational planning carries real economic weight. The honest counterpoint is that SDDP remained the more cost-effective approach under the conditions tested. The hybrid method narrowed the gap when the inflow model was relaxed, but it did not close it.

Several research directions look promising to further explore the potential advantages of DRL. Parallelizing the inner optimization across scenarios would improve training efficiency, particularly when the per-stage solve is expensive, for example when richer network or operational constraints are introduced. More recent RL algorithms such as Twin Delayed DDPG and Soft Actor-Critic address some of the stability issues of DDPG and could improve sample efficiency. And because we already have an explicit model inside the optimization layer, model-based RL (the family of algorithms behind systems such as AlphaZero) is a natural fit: the agent could evaluate several candidate target volumes by solving the single-stage problem for each, and use that information to plan ahead more effectively.

DRL itself is a developing toolset rather than a finished product. The field has advanced substantially in the past few years, with more stable training algorithms, better sample efficiency, and active research on scaling to the kinds of large state and action spaces characteristic of power systems. For long-horizon planning problems such as hydrothermal dispatch, we believe the most likely path forward is hybrid: optimization handles what it does best, namely enforcing physical constraints and producing tractable per-stage decisions, while DRL handles the temporal coupling and the parts of the problem that resist convex modeling.

References

Pereira, M.V.F. and Pinto, L.M.V.G. (1991). Multi-stage stochastic optimization applied to energy planning. Mathematical Programming, 52(1), 359–375.

Rosemberg, A.W., Street, A., Garcia, J.D., Valladão, D.M., Silva, T. and Dowson, O. (2022). Assessing the cost of network simplifications in long-term hydrothermal dispatch planning models. IEEE Transactions on Sustainable Energy, 13(1), 196–206.

Lillicrap, T.P. et al. (2015). Continuous control with deep reinforcement learning. arXiv:1509.02971.

Lei, L., Tan, Y., Dahlenburg, G., Xiang, W. and Zheng, K. (2021). Dynamic energy dispatch based on deep reinforcement learning in IoT-driven smart isolated microgrids. IEEE Internet of Things Journal, 8(10), 7938–7953.

Sivamayil, K., Rajasekar, E., Aljafari, B., Nikolovski, S., Vairavasundaram, S. and Vairavasundaram, I. (2023). A systematic study on reinforcement learning based applications. Energies, 16(3).

inSight

HOW AI REASONING UNLOCKED THE NEXT GENERATION OF SOFTWARE ENGINEERING

Raphael Sampaio

A new generation of large language models with explicit reasoning capabilities has changed what is possible in software development. Earlier code-completion tools relied on statistical pattern matching and were useful primarily for short, local suggestions. The latest reasoning models can decompose engineering problems into intermediate steps, plan a sequence of actions, inspect their own work, and iterate toward a solution. When this reasoning capability is embedded in an appropriate execution environment, an entirely different practice becomes available, one in which the human engineer is no longer the sole producer of code but the architect of a process that includes an autonomous coding agent.

From the Transformer to reasoning models

The path that leads to today's coding agents has earlier roots in decades of work on neural language modeling, recurrent architectures, and prior attention mechanisms, but its decisive milestone came in 2017 with the publication of "Attention Is All You Need" by Vaswani and colleagues, most of them at Google Brain and Google Research. The paper introduced the Transformer, a neural-network architecture that replaced the recurrent and convolutional layers then dominant in language modeling with a mechanism based entirely on self-attention. The Transformer turned out to be unusually well-suited to scaling, in both parameter count and training data, and it has remained the substrate of essentially every large language model released since.

The first wave of practical implications appeared in 2018 with GPT-1, OpenAI's generative pre-training approach, and BERT, Google's bidirectional encoder. Both established the now-dominant paradigm of pre-training on large corpora of unlabeled text and then fine-tuning on narrower tasks. The differences between the two designs (decoder-only versus encoder-only, autoregressive versus masked) shaped the divergent application paths that followed.

The 2019–2020 period made scaling itself the central question. GPT-2 demonstrated that a moderately larger model trained on more text produced qualitatively better text. GPT-3, with 175 billion parameters, showed in 2020 that further scaling unlocked in-context learning, the ability to perform new tasks given only a handful of examples in the prompt. The scaling laws published the same year by Kaplan and colleagues offered a quantitative framework for how loss decreased with parameters, data, and compute, and they shaped much of the field's subsequent investment.

The 2022 turning point was alignment rather than capability. InstructGPT and the closely related techniques behind ChatGPT (released in November 2022) used reinforcement learning from human feedback (RLHF) to make models follow instructions reliably and refuse plainly inappropriate requests. The technique fine-tunes a pre-trained model through a reward model that learns from human preferences over candidate completions, shaping the model's outputs toward responses humans' rate as helpful, honest, and safe. The technical advance was modest in absolute terms; the user-experience advance was decisive. Within months, the underlying technology had reached a mass audience.

The 2023–2024 period saw the proliferation of providers and the rise of multimodality. GPT-4, Anthropic's Claude, Google's Gemini, and Meta's Llama family (the latter as open weights) all reached general

availability, with most adding image, audio, or video understanding. Code-completion tools that had existed at smaller scales since 2021, most notably GitHub Copilot, gained GPT-4-class chat features (Copilot Chat moved to GPT-4 in late 2023), and the quality of assistance improved correspondingly.

The next axis appeared in late 2024 with OpenAI's o1 family of reasoning models, followed in early 2025 by DeepSeek's R1, the first widely used open-weights reasoning model, and later that year by OpenAI's o3 family. The technical idea was that a model could be given more compute at the time of answering rather than only at training time and could spend that compute on chain-of-thought reasoning, self-checking, and iterative refinement. For tasks involving multi-step planning, mathematics, and code, the gains were substantial. By 2025 the leading model families across providers all included reasoning variants. Anthropic folded extended thinking into the Claude family, OpenAI shipped the GPT-5 generation as a reasoning-first family, and reasoning had become a baseline expectation rather than a frontier feature.

The combination of capable reasoning models and the engineering needed to give them tools, file-system access, and execution environments produced the wave of agentic coding tools that define the current period. Through 2025, a new class of command-line agents reached general availability, including GitHub Copilot Agent, Gemini CLI, and Anthropic's Claude Code, while editor-integrated tools, such as Cursor, extended the same pattern into the IDE. Within months, the unit of interaction had moved from the line of code to the engineering task.

Era	Core Breakthrough	Primary Value in Software Engineering
2017–21	Transformers and Scaling	Statistical pattern matching and line-by-line completion
2022–23	Alignment (RLHF) and Chat	Conversational assistance, explaining code
2024–25	Reasoning at Inference	Multistep planning, chain-of-thought, self-correction
2025–	Agentic Scaffolding (Harness)	Autonomous file editing, CLI execution, Model Context Protocol (MCP) integration

The narrative arc is consistent across these milestones: architecture (the Transformer) made scaling possible; scaling produced capability; alignment made the capability accessible; reasoning extended the capability into multistep tasks; and tooling turned the resulting models into autonomous engineering agents. Each turning point opened the door to the next.

Foundation models and their constraints

Underneath every coding agent lies a foundation model. The major providers (Anthropic with Claude, OpenAI with GPT, Google with Gemini, plus a growing set of open-weight families such as Kimi K2 and Qwen 3) all follow a similar deployment pattern: the model is trained once and then served with frozen parameters. This pattern carries an important consequence: the model itself does not learn from one session to the next. It cannot form new permanent memories, and every new conversation begins with a blank slate. What the model "remembers" within a session is limited to the contents of its context window, the bounded sequence of tokens that includes the system prompt, the user's instructions, prior exchanges, tool outputs, and the model's own intermediate reasoning.

Each provider typically offers its models in tiers that balance capability, latency, and cost. While specific vendor branding shifts rapidly, the underlying pattern remains an industry standard: a flagship model for the hardest reasoning tasks (e.g., Anthropic's Opus, OpenAI's GPT-5, Google's Gemini Pro), a balanced everyday model (Sonnet, GPT-5 mini, Gemini Flash), and a fast lightweight model (Haiku, GPT-5 nano,

Gemini Nano). The choice of tier is a context-engineering decision, since heavier reasoning is rarely needed at every step of an engineering task and matching the tier to the difficulty of the subtask is a meaningful lever on quality, latency, and cost.

Modern context windows are large by historical standards. The most recent Gemini and GPT releases reach roughly one million tokens, the latest Claude generation reaches comparable lengths in its extended-context configurations, and open-weight models such as Kimi and Qwen sit in the low hundreds of thousands. They remain finite, however, and two related phenomena make their management non-trivial. The first is the simple truncation that occurs when the window fills: once the limit is reached, the oldest content is discarded, and the model loses access to it. The second, more subtle, is context rot: as the window approaches its capacity, accuracy degrades and the model begins to lose track of details even within content, still nominally present. Empirical benchmarks of long-context retrieval consistently show that more context is not always better. Recently, the introduction of prompt caching has mitigated some of these friction points by allowing models to recall massive codebases and system prompts cheaply and rapidly across turns, though it does not solve the fundamental limits of working memory.

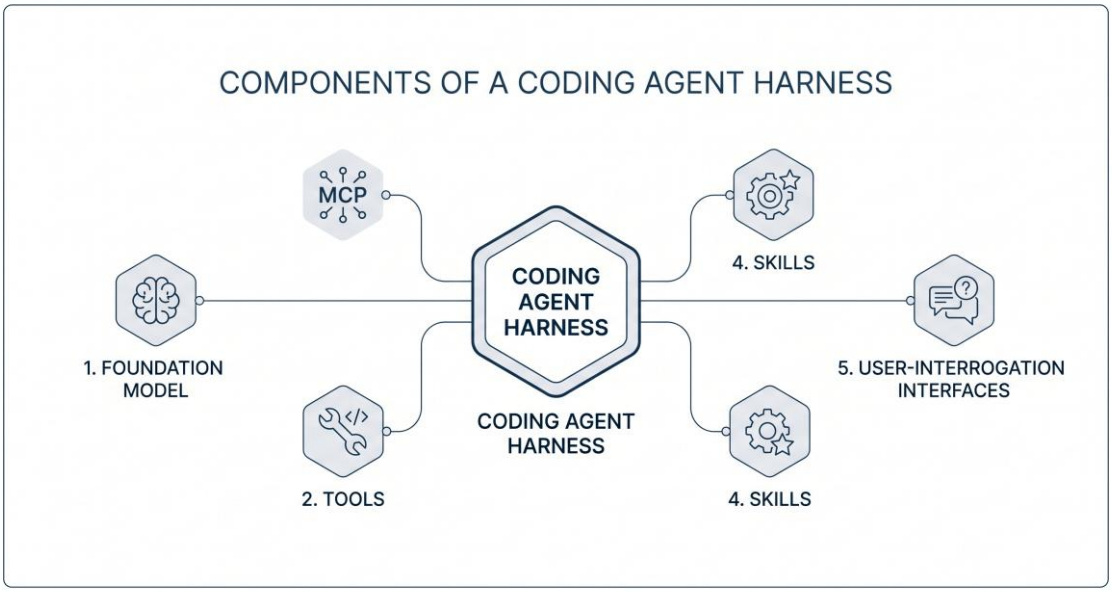
These constraints matter operationally. The reasoning model has the capability to plan, write, and verify code, but it has neither persistent memory across sessions nor unlimited working memory within a session. The quality of any extended interaction therefore depends critically on what is placed inside the context window and what is kept out.

The agent's harness

A coding agent is more than the foundation model that powers it. It is a harness that gives model-controlled access to a set of tools, a structured way of recording project knowledge, and a discipline for managing the boundary between planning and execution. Several agents now compete in this category, including GitHub Copilot Agent, Gemini CLI, Claude Code, Cursor, and Cline, and the architecture they share is broadly similar even where the surface vocabulary differs.

The agent operates inside a project directory and is equipped with a small set of fundamental tools: a file editor, a shell for running tests and builds, a web search and fetch capability, and a programmatic tool-calling interface. These primitives allow the agent to read source files, modify them in place, run the test suite, and consult external documentation, all without the engineer leaving the terminal. Two further mechanisms extend this baseline. The Model Context Protocol (MCP) is an open standard, released by Anthropic in late 2024, that allows agents to connect to external systems through a uniform interface, in much the same way that USB-C unified the proliferation of cables before it. Although Anthropic-authored, MCP is now adopted across a range of hosts that include Cursor, Cline, and several editor-integrated agents, which makes it a genuine portability layer rather than a vendor-specific protocol. A representative example is Context7, an MCP server that pulls updated, version-specific documentation for libraries and packages, allowing the agent to write against the API in current use rather than the API it happened to see during training. Skills are modular text-based capabilities, packaged as instructions and supporting resources, that the agent activates automatically when their description matches the task at hand.

A further element of recent harnesses is the introduction of structured user-interrogation tools, mature implementations of which (such as Claude Code's AskUserQuestion) let the agent pause and put a multi-choice question to the engineer rather than guess at an ambiguous requirement. The mechanism is small but consequential, because the most frequent source of wasted work in agentic coding is the model proceeding confidently down the wrong interpretation of an instruction.



Coding-agent harness — foundation model, tools, MCP servers, skills, and user-interrogation interfaces.

Two harness features deserve particular attention because they directly address the memory and discipline problems described in the previous section. The first is the project memory file, a markdown document placed at the root of the project that the agent reads at the start of every session. The convention has converged across tools under different names: CLAUDE.md for Claude Code, GEMINI.md for Gemini CLI, AGENTS.md as a more recent cross-tool convention adopted by Cursor, GitHub Copilot, OpenAI Codex, and others. The file records the project overview, the conventions to follow, the commands that build and test the code, and any context that would otherwise have to be rediscovered each time. It is, in effect, the persistent memory that the model itself lacks. A well-maintained project memory file transforms what would be a stream of identical re-explorations into a series of focused interactions that begin with the relevant context already loaded.

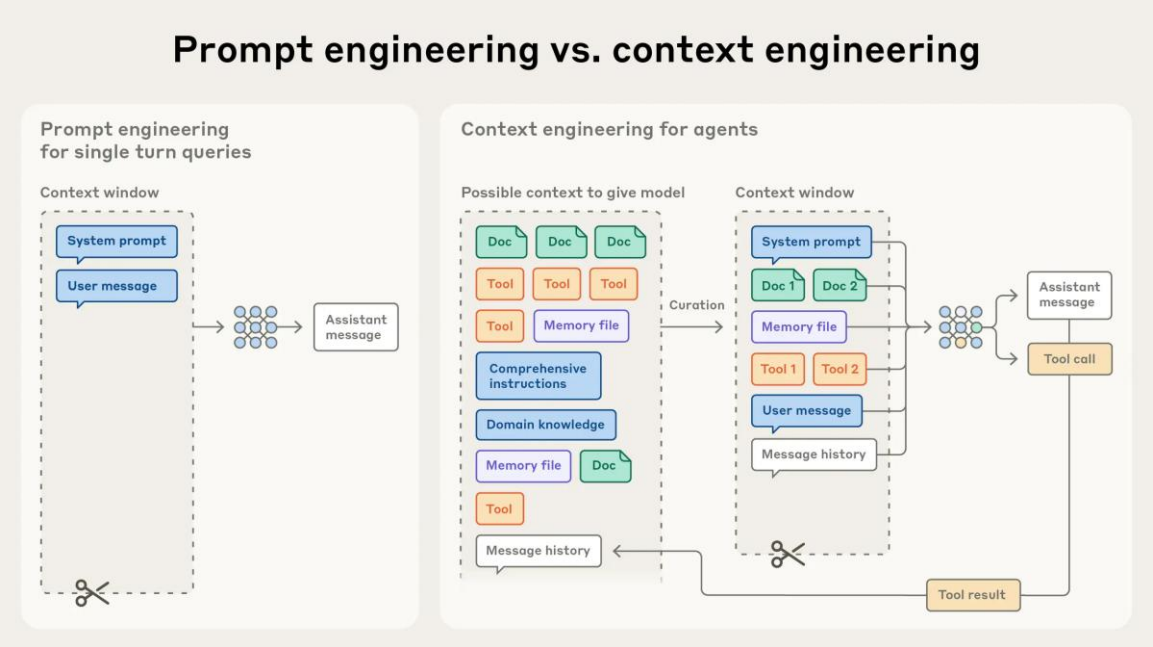
The second is the discipline of separating planning from execution, implemented under various names across the tooling. In this mode the agent is prevented from editing files or running commands with side effects; it can only read, analyze, and propose. The output is a written plan that the engineer reviews and approves before any code changes occur. This separation prevents the agent from committing to an architectural direction that the human disagrees with, eliminates wasted work on premature implementations, and prevents the model from falling into an “agentic doom loop”, a common failure mode where a reasoning model repeatedly writes failing tests, attempts to fix them, and fails again without human intervention. The principle is straightforward: an autonomous agent should not be turned loose on a challenging task without a reviewed and approved plan.

Beyond vibe coding: prompt and context engineering

The accessibility of these tools has produced a contrasting development style sometimes referred to as vibe coding, a term coined by Andrej Karpathy in early 2025 to describe, in his phrase, a workflow in which the user “fully gives in to the vibes,” accepts the model’s diffs without reading them, pastes errors back without comment, and lets the code grow beyond their own comprehension. For exploration scripts and disposable prototypes, the approach is genuinely useful, and it has lowered the barrier to producing working software for users with limited programming background. For software that must be maintained, audited, and trusted in production, accept-all generation produces code that is difficult to

reason about, often fragile under edge cases, and exposed to security weaknesses that no one ever read closely enough to catch. The reputation of AI-assisted coding has suffered accordingly.

The professional response to this failure mode is the discipline known as prompt and context engineering. Prompt engineering concerns the formulation of the instructions given to the model: their specificity, their structure, the explicit boundaries they establish, and the criteria by which success will be judged. Context engineering, the broader practice, concerns the curation of everything the model sees inside its context window during a task, including the system prompt, the project memory file, the relevant source files, the prior conversation, the tool definitions, and the outputs of tools already invoked. The complexity is not eliminated; it is moved out of the moment-to-moment interaction and into a deliberately engineered environment that the agent can rely on.



Prompt engineering vs. context engineering
(anthropic.com/engineering/effective-context-engineering-for-ai-agents)

The practical implications are concrete. A request to refactor a module is not a single sentence typed into a chat box; it is a small project that begins with the agent reading the project memory file, continues with a review of the relevant source, produces a plan that names the proposed changes and the tests that will validate them, and only then executes the plan in stages with each stage verified before the next begins. Tests and code coverage, far from being secondary concerns, become the safety net that allows iteration to remain fearless. AI-generated code without rigorous coverage is a maintenance liability waiting to surface; AI-generated code accompanied by a strong test suite can be modified, extended, and debugged with confidence.

From a planning perspective, this shift reframes the engineer's role. The most valuable contribution is no longer the volume of code written but the quality of the context provided, the precision of the plans approved, and the rigor of the verification applied. The model supplies reasoning capacity at a scale that no individual engineer can match, the engineer supplies judgment, domain knowledge, and accountability for the result.

Conclusions

Eight years separate "Attention Is All You Need" from the autonomous coding agents now in everyday use. The path runs through the Transformer architecture, the scaling-laws era, the alignment

breakthroughs that made instruction-following reliable, the reasoning models that extended capability into multi-step planning, and the agentic tooling that put all of it inside the engineer's workspace. Reasoning models capable of multi-step planning, when paired with a well-designed harness that exposes tools, structures project memory, and separates planning from execution, make it possible to take on larger engineering tasks than was practical only a few years ago. The constraints of these models, in particular the absence of persistent memory across sessions and the finite context window, are real but manageable through disciplined engineering practice.

The promise of these tools is significant, but only when realized through prompt and context engineering is disciplined enough to be trustworthy. For organizations whose analytical models and tooling underpin decisions that matter, the implication is direct: the coming years will reward those that invest in this craft as seriously as they have invested in the craft of modeling itself.

inBrief

EXPANDING THE SDDP PLATFORM EXPERIENCE THROUGH AI: CHATBOTS, MCPS, AND AI REASONING

Carolina Abdu, David Parrini and Ricardo Perez

Imagine two people trying to reach the same destination. One has an old car with a paper map; the other, a modern vehicle with integrated systems, real-time navigation, route optimization and intelligent assistance. Both are still driving, and both remain responsible for their decisions. But one of them will move faster, react quicker, process more information and reach the destination far more efficiently. That, in essence, is what Artificial Intelligence represents today. As Sam Altman, OpenAI's CEO, has put it: "AI will not replace humans, but humans using AI will replace those who don't."

But the great car only improves your travel experience; it does not turn you into a good driver. To really take advantage of AI, the Anthropic AI Fluency Course defines four steps that matter. The first is delegation: you must know where you want to go and understand what AI can do well. The second is description: you must be able to transmit that intention to the AI, giving it the role to play, the context it should use and the rules it must follow. If you cannot configure the car's system and type the destination into the GPS, the supercar will not help you. Once you start driving, you need discernment: the AI will suggest possibilities, and you remain responsible for checking that the route it picked is actually the right one for you. How many times has the GPS chosen a fast route full of tolls, or one that simply does not feel safe? With AI, it is the same. The last step is diligence: never sign your name on something you cannot explain. You can delegate a task, but never the understanding behind it. In our analogy, you can switch on the autopilot, but you should not trust it so completely that you would no longer know how to take back the wheel when something goes wrong.

In short, the four Ds form a chain. Delegation is about knowing where AI helps and where it does not. Description is about giving it the role, the context and the rules it needs to do the job. These two come before the model is used. The other two come after: discernment is checking that the route the AI chose is the right one — that the approach makes sense and the output is actually useful and correct — and diligence is the awareness that, however good the assistant, the signature on the final answer is yours. Throughout this article, as we walk through the ways AI now connects to the PSR models, these four steps are worth keeping in mind: they are what separates using AI well from merely using it.

AI's traditional use

The first step away from the paper map was the rise of the LLMs. The earliest way of using them was the simplest one: write a message (a prompt), send it to the model and read the answer that comes back. How does this actually work? LLMs (Large Language Models) are trained on enormous text datasets. The words are first tokenized, i.e., broken into small pieces and converted into numerical IDs the model can process. From that training, the model learns the probability that a given word or phrase should follow another in a given context and uses those probabilities to generate new content on demand. That is what we call generative AI. It produces emails, summaries, code snippets, and drafts of reports. The result is genuinely useful, but it does have limits.

A generic LLM is constrained by the dataset it was trained on, and retraining a model is expensive enough that it cannot be done in real time. The consequence is a lack of context. If you ask the AI about something specific to PSR that was not part of its training, it does not admit it; it guesses, and what is worse, it does

so with complete confidence. Ask it, for instance, “what is a renewable station in SDDP?” and the model will answer using public information about the physical concept of renewable plants, something like: “A Renewable Station is a power generation facility that produces electricity from naturally replenished energy sources such as solar, wind, hydro, or biomass.” This is what we call hallucination.

Grounded intelligence: an assistant that knows PSR

To close that lack of context, the LLM must somehow access PSR’s own knowledge. Since fully retraining the model with PSR documentation would be expensive, imagine instead that we can plug a pen-drive into the LLM at the moment it needs to be answered. The technique that does precisely that is Retrieval-Augmented Generation (RAG). It works in four small steps. First, the PSR documentation (manuals, methodology notes, examples) is split into smaller pieces called chunks, tokenized and embedded into a vector index, so that it speaks the same numeric language the LLM uses to generate text (0). When a user asks a question (1), the question itself is also vectorized, and a similarity search using cosine distance returns the passages whose meaning is closest to it. Those passages are injected into the prompt as context, so the LLM receives both the user’s question and the documentation it needs to answer it correctly — exactly the description step we talked about in the opening paragraph (2). The LLM then composes a grounded answer from that evidence, not from its general training (3).

The PSR Assistant uses exactly this technique to let users ask specific questions about PSR. It does not only answer the question; it also returns the link to the relevant page in the PSR Knowledge Hub , so the user can double-check the information. Beyond avoiding hallucinations, the real gain is time: instead of searching across hundreds of pages, the user gets a direct, sourced answer. The Assistant has two siblings built on the same pattern, but grounded on the source code of PSR libraries— one for PSR Factory , our Python automation API, and one for PSRIO, our BI tool. For users who are not yet fluent in those libraries, they short-cut the blank-page problem: ask for a typical script and get a working starting point you can adapt.

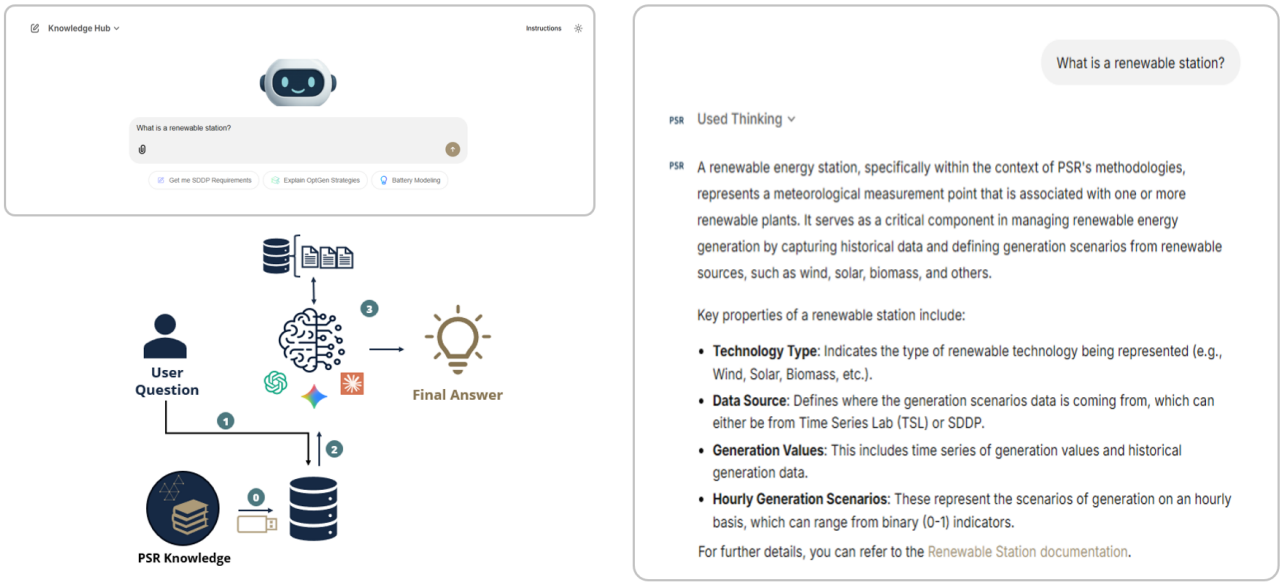


Figure 1 — RAG anchors the LLM on PSR’s own documentation, replacing hallucinated definitions with grounded, sourced answers.

Agentic AI: from answers to actions

But what if AI could go beyond generating content and start executing tasks? Imagine that the manual work an analyst performs every day — discovering the input data of a model, editing that data, running

cases on-premises or on the cloud, visualizing the results — could be turned into automated tasks triggered by a single request in natural language. This is the third step in the journey: from generative AI to agentic AI, a system designed to execute complex workflows by combining autonomous planning, long-term memory and the ability to interact with external applications. But if the LLM still has limited data, how can it possibly know about our data?

Modern AI platforms — OpenAI's, Anthropic's and others — now support customized branches that give the model access to external applications. The model then behaves as an orchestrator: it uses those branches, together with its own LLM, to execute tasks for the user. To execute a task automatically, the model needs access to a customized code function that it can call, and that can in turn reach external data sources. The standard way to make this connection is the Model Context Protocol (MCP), an open protocol that lets any LLM interact with external systems through a consistent interface. An MCP server publishes a catalogue of tools — typed functions, each with a clear description — that the model can browse and invoke. The PSR Agent is our MCP server. It exposes functions that, through the PSR Factory Python API, talk to PSR models databases. Orchestration usually begins with an internal plan of what should be done; the model then compares this plan against the descriptions in the tool catalogue, by similarity, and selects the tool that best matches the task at hand.

For the AI to perform all of that well, it needs a good description of how these tools works and which workflow should be used to each task. But it would be tedious to write those instructions from scratch every time. The natural solution is to package them once and reuse them. This is the idea behind Skills: bundles of instructions, examples and helper scripts that the agent loads on demand when it recognizes a familiar task type. Where MCP gives the AI access, Skills give it expertise — the right order in which to call the tools, the common pitfalls to avoid, the PSR house style for reporting results. A typical workflow is illustrated below; at the end of it, the LLM is used once more, this time to write up the result or to report a success or failure message.

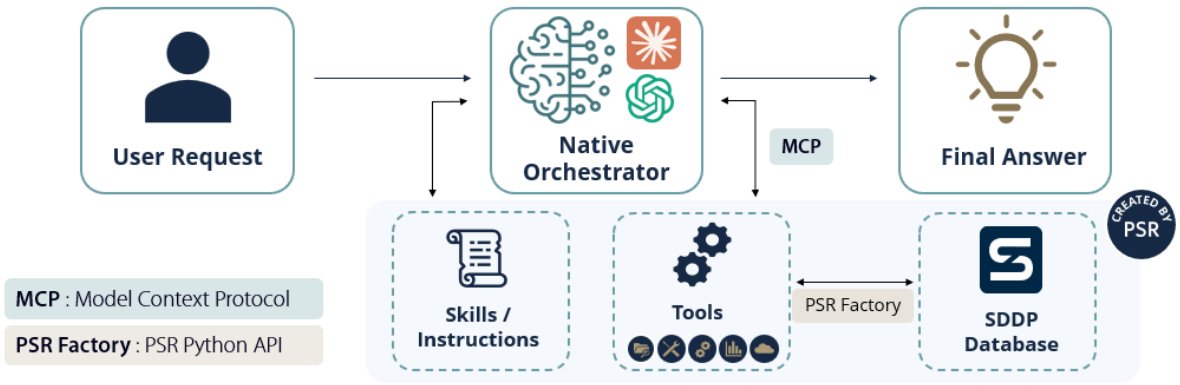


Figure 2 — The PSR Agent: an AI model uses its LLM to discover and load Skills for domain expertise, accesses PSR tools through MCP, and reads from and writes to the SDDP database via PSR Factory.

The toolbox PSR built into the Agent is shaped around the parts of the day that take the longest. Tracking down an object, navigating its relationships or filtering elements by condition becomes a single question, rather than a sequence of tabs and clicks through the interface. Repetitive bulk operations — updating capacities, scaling time series, creating new objects — can be executed automatically in a safe-copy mode that preserves the original case. Simulations in SDDP, OptGen and NCP can be launched directly from the conversation, with the Agent monitoring execution status and extracting structured summaries of errors, warnings and convergence information from the run logs. Scenario comparisons, revenue calculations and result analyses can also be performed through the conversation; a single instruction is enough to package the outputs and notify the team.

In summary, the current set of tools was designed to reduce the repetitive operational work traditionally performed through the graphical interface. The project is intended to evolve incrementally: as new recurring workflows and user needs are identified, additional capabilities will be incorporated into the Agent over time.

In practice, a single prompt such as "update Plant X's installed capacity to 1,350 MW, run SDDP for the 2026 horizon, and email the results to the operations team" already maps to a workflow that previously required several manual steps across different interfaces. The user still reviews the proposed actions and validates the final answer, but the operational friction drops sharply.

The next step: a reasoning layer

Grounding the LLM in PSR knowledge and connecting it to PSR tools covers the "information" and "execution" parts of the problem. The third piece, still in development, is the analysis layer: an agent that does not only answer questions or run operational tasks on request, but interprets simulation results, forms hypotheses about what is driving them, and pulls the data needed to test those hypotheses.

The prototype, currently under development, is structured as a multi-agent pipeline. An orchestrator dispatches the user request to specialized sub-agents — one for problem classification, one for cause hypothesis, one for data discovery — each with its own narrow set of tools. The orchestrator then combines their outputs into a single synthesized answer, with traceable evidence the analyst can audit step by step.

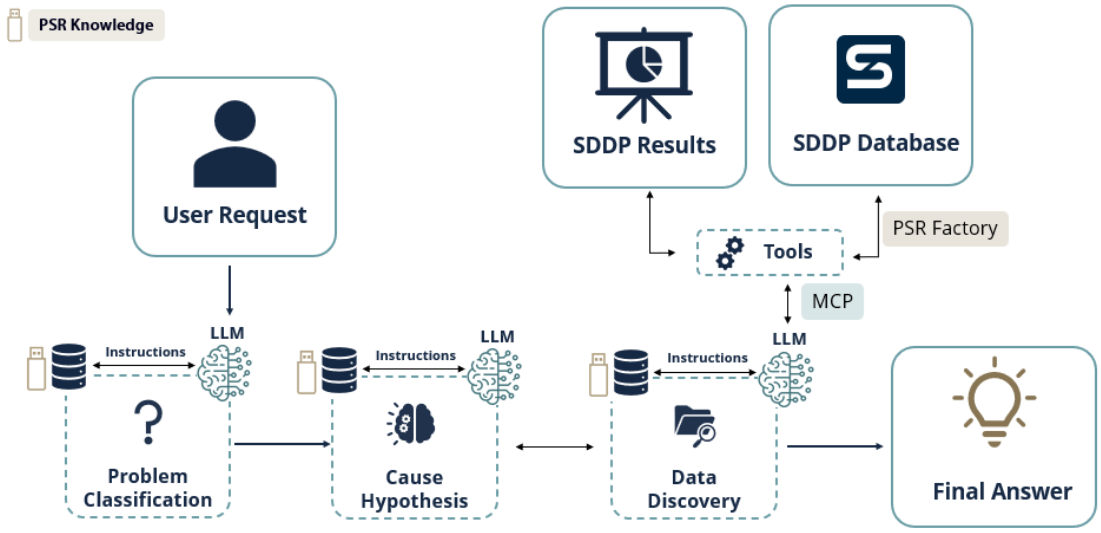


Figure 3 — Reasoning agent under development: specialized sub-agents classify the problem, generate hypotheses and discover data, then return a synthesized, evidence-backed analytical answer.

Taken together, RAG, agentic AI and the reasoning layer trace a clear path for AI inside PSR: from explaining the models, to operating them, to reasoning over their results. The goal is not to replace the analyst, but to take the routine work off the analyst's plate, so that the time saved goes to what really needs human judgement.

inDepth

AI REASONING IN MATH AND CODING: A STOCHASTIC GENERATION-EXPANSION SOLVER

Rafael Benchimol Klausner

Introduction

AI reasoning in mathematics

For most of the deep-learning era, the relationship between large language models and mathematics was uneasy: systems that excelled at natural-language generation remained unreliable on multi-step reasoning and formal mathematics. That changed rapidly after 2024, as frontier systems reached Olympiad-level performance and began integrating directly with formal verification environments such as Lean.

More interesting than the benchmark progression itself, however, is what this capability means for working mathematicians. Terence Tao has become one of the clearest voices articulating the emerging role of AI in mathematical research. In a January 2025 article in the *Notices of the American Mathematical Society* on machine-assisted proof, and later through the release of a flexible formalization of his textbook *Analysis I* in Lean, Tao argued not that language models had overcome mathematicians, but that they had become useful collaborators. Their value lies in reducing the overhead surrounding research: searching literature, checking standard arguments, formalizing definitions, writing small computational experiments, and testing whether an approach is worth pursuing. In 2023, Tao had already used ChatGPT alongside Lean in work related to formalizing aspects of a proof of the Polynomial Freiman–Ruzsa conjecture.

The relevant shift is therefore not merely that AI systems can solve harder mathematical problems, but that they are increasingly capable of operating across formal reasoning, symbolic manipulation, code generation, and verification within a unified workflow.

AI reasoning in code

The trajectory in software engineering is structurally similar to the one in mathematics, although more visible because its outputs are merged code rather than written proofs. The shift is from suggestion-style autocomplete toward agentic systems that inspect repositories, plan edits across multiple files, execute tests, and iterate on failures. Benchmarks such as SWE-bench Verified now evaluate these systems on real GitHub issues drawn from production repositories, and by 2026 frontier agents were already resolving a substantial fraction of repository-level software engineering tasks on previously unseen codebases.

Beyond benchmarks, the industrial signal is increasingly difficult to ignore. By 2026, coding agents had become embedded in the daily workflows of major engineering organizations for refactoring, debugging, migration, and test generation tasks. Anthropic executives stated publicly that Claude was already writing most of the company's internal code, while Google's 2025 DORA State of AI-Assisted Software Development report found AI adoption broadly associated with higher engineering throughput, with productivity depending as much on testing infrastructure and operational discipline as on the specific model used.

Read together, the trajectories in AI for mathematics and AI for code are converging. When supplied with both a research paper and a codebase, frontier systems can participate in a coupled reasoning-and-implementation loop: interpreting algorithms, identifying correctness conditions, translating them into production code, generating regression tests, and iterating through execution and verification. The remainder of this article reports a deliberate experiment in exactly such a workflow, with the human setting high-level objectives, approving implementation strategies, and intervening mainly when tests exposed algorithmic or convergence failures.

The application: stochastic generation expansion

LightBDMM.jl is a PSR Julia package developed against this environment. It implements the accelerated Benders Decomposition with Multiple Masters (a-BDMM) algorithm that was introduced in the article: *An Integrated Progressive Hedging and Benders Decomposition with Multiple Master Method to Solve the Brazilian Generation Expansion Problem* (A. Soares, A. Street, T. Andrade, and J. D. Garcia). The package structure, implementation, regression tests, documentation, and continuous-integration setup were generated by Claude Code, then iteratively corrected through execution, testing, debugging, and human review. Subsequent extensions — mixed-integer (MIP) first-stage decisions with quadratic-penalty linearization, and distributed execution over MPI were added by the same assistant.

The project was deliberately set up as a low-interaction experiment: how far can a frontier agentic system carry an applied mathematics implementation when the human author confines themselves to high-level objectives and end-state review, rather than per-step steering? Correctness during the cycle was anchored not on human review of intermediate steps, but on an independent ground-truth validation harness, which is described later.

We already have implementations of Benders decomposition and Progressive Hedging in separate repositories with MPI parallelization and mixed-integer representation. These repositories were supplied to the coding agent as reference implementations for architectural patterns, testing procedures, and documentation structure.

The mathematical structure underneath stochastic generation expansion is the two-stage stochastic program: an investment decision today (which units to build, which lines to reinforce, which contracts to sign) is made before next year's uncertainty (demand, hydrology, fuel prices, renewable output) is observed, an operational recourse decision is then taken once each scenario has materialized.

The math-reasoning loop

The interesting aspect of this implementation was not symbolic mathematics or automated theorem proving. The challenge was translating a research paper into a working optimization package: understanding the algorithmic structure, planning the implementation, identifying convergence challenges, and iteratively correcting them through testing and validation. This experiment was only meaningful because correctness could be checked independently: every implementation was continuously validated against deterministic-equivalent solutions and benchmark behavior rather than judged only by whether the generated code appeared plausible.

The a-BDMM algorithm augments classical Benders along two axes. Multiple masters replace the single aggregated master with one master per scenario, each anchored on a different recourse problem and sharing a common cut pool, producing a tighter recourse approximation in fewer iterations. Progressive Hedging acceleration adds a quadratic consensus penalty that drives the per-scenario decisions toward a common value.

What makes the algorithm a useful test of mathematical AI reasoning is that several of its correctness conditions are not obvious from the headline description and would be easy to miss in a naive re-implementation:

1. A separate LP for the lower bound after PH activates. Once the consensus penalty is on, the algorithm's lower bound must be computed from a master that contains only the Lagrangian term $w_s(x_s - x)$ and not the quadratic penalty. The implementation maintains a second pool of lower-bound masters for exactly this purpose.
2. If PH is initialized at a degenerate point, for example, with all components of x set to zero, the algorithm may converge prematurely to a consensus solution that is nevertheless suboptimal.

Working through these issues required an iterative planning and debugging process. Before writing code, the assistant generated implementation plans, surfaced convergence risks, and proposed stabilization strategies. These observations still required human verification but surfacing them early reduced the likelihood of propagating conceptual mistakes into the implementation itself.

The coding loop

The coding loop was the second stage of the process and followed the same low-interaction protocol. Claude Code generated the package structure, implementation, regression tests, documentation, and continuous integration setup, while the human author provided only high-level specifications at the beginning of each cycle and reviewed the final outputs. The important point was not that the first implementation was correct, but that the agent could keep the repository, algorithm, and test feedback in context while iterating toward a validated implementation.

The initial implementation already reproduced most of the algorithmic structure correctly, including the Benders decomposition workflow and the benchmark newsvendor test problem. The first major issue appeared in the cut aggregation logic: the implementation initially produced incorrect objective values because scenario contributions were not properly combined. After correcting the aggregation and weighting of cuts, the BDMM variant converged quickly and produced the expected solution.

The more interesting failure emerged in the accelerated a-BDMM variant. All state variables were initialized at zero, and the Progressive Hedging penalty terms immediately pushed the algorithm toward consensus around this uninformed initial point. As a result, the method became trapped in a suboptimal “do nothing” solution and failed to improve. The eventual fix was to introduce a warmup phase in which the decomposition explored freely for several iterations before activating the consensus penalties. Once implemented, this stabilization strategy resolved the convergence issue on benchmark problems and matched the intuition that agreement mechanisms should not dominate before informative cuts have been generated.

The lower-bound logic, however, required a much more interactive debugging process. Once the PH quadratic penalties became active, the bound calculation no longer behaved consistently. Several implementation attempts failed in different ways: using the upper bound as a proxy produced artificial convergence in the first iteration; using the raw master objectives caused oscillatory lower bounds after PH activation; and subtracting the quadratic penalties directly sometimes generated invalid states with LB >UB.

Unlike the warmup mechanism, which emerged with relatively little human intervention, the lower-bound phase required repeated interaction between the human author and the assistant. The debugging process eventually became less about software engineering and more about computational interpretation of the algorithm itself: determining which quantities remained valid lower bounds after PH activation, how convergence should be measured, and which parts of the augmented objective were

optimization artifacts rather than components of the original stochastic program. The final implementation eventually recovered a correct lower-bound procedure described on the paper.

Overall, the debugging process was notable because most failures were not syntactic or software-engineering issues, but algorithmic challenges related to convergence behavior and the interaction between decomposition and consensus mechanisms.

Validation on a generation-expansion model

The solver was exercised on a stochastic generation expansion problem — fifteen candidate plants, forty-five industrial customers, and fifty stochastic activation scenarios. Each method was run to a 0.1–% relative gap at the same tolerance.

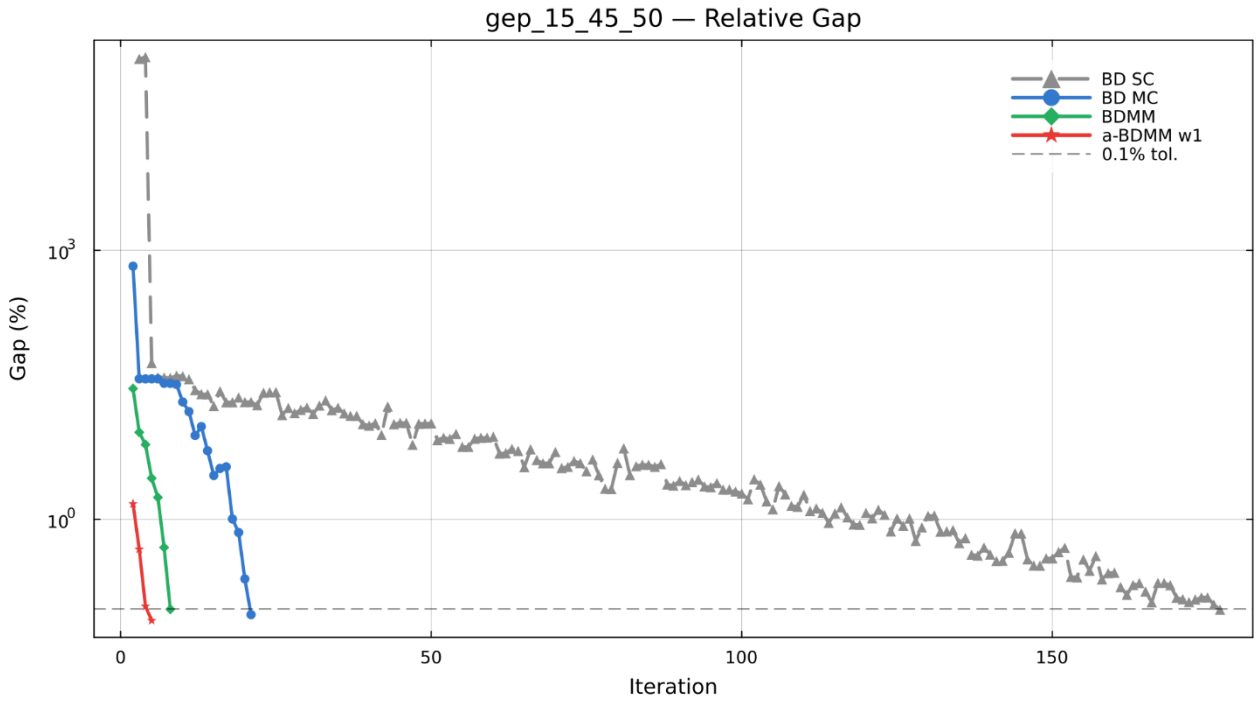


Figure 1 - Relative optimality gap as a function of iteration count for each algorithm on the generation-expansion testbed. a-BDMM collapses the gap in 5 iterations, against 8 for plain BDMM, 21 for multi-cut Benders, and 177 for single-cut Benders.

The qualitative result of interest is the iteration count. a-BDMM closes the bound gap in 5 iterations, against 177 iterations for classical single-cut Benders on the same instance — a roughly 35-fold reduction. Plain BDMM (without the Progressive Hedging acceleration) lies in between, at 8 iterations, isolating the contribution of the consensus penalty.

The validation harness

In a low-interaction development protocol, the validation harness is not merely a quality-assurance tool; it is the mechanism that makes the workflow viable. Every regression test in the suite is solved both by LightBDMM.jl and by the monolithic deterministic equivalent — the exact extensive-form formulation of the stochastic program used as an independent reference solution. The results are required to match every benchmark instance, so the implementation is continuously checked against an external ground truth rather than only against its own internal logic. This serves as the numerical counterpart of the earlier reasoning loop: conditions identified during planning and debugging are revalidated computationally against a benchmark that shares none of LightBDMM.jl's implementation. Any

conceptual or implementation drift introduced during the agentic coding cycle would therefore appear as a deterministic-equivalent mismatch before reaching production.

The full regression suite runs in continuous integration on every proposed code change. At the time of writing, the package maintains approximately 98% automated code coverage.

Outlook

The experiment does not demonstrate autonomous mathematical engineering. It demonstrates something narrower but operationally important: when algorithms have strong mathematical structure, benchmark behavior, and independent validation procedures, agentic coding systems can substantially reduce the engineering cost of translating research methods into production implementations.

Two-stage stochastic optimization is one of the oldest and best-understood frameworks in operations research, and a-BDMM itself has existed in the literature since 2021. The relevant change is therefore not the novelty of the algorithm, but the reduction in the cost and time required to move from paper to tested software. In this case, the combination of mathematical structure, deterministic-equivalent validation, and AI-assisted development made it possible to build a production-quality implementation including decomposition variants, regression tests, benchmarking infrastructure, continuous integration, and documentation, all within a single development cycle.

More broadly, mathematically structured domains with independent reference solutions appear especially compatible with agentic development workflows. When large parts of the translation from paper to production can be accelerated and continuously validated, organizations gain the ability to experiment with, operationalize, and deploy a much larger fraction of the optimization literature than was previously practical.

inDepth

ASSESSING THE CAPABILITY OF AGENTIC AI IN A STOCHASTIC HYDROTHERMAL SCHEDULING CASE STUDY

Julio Alberto Silva Dias and Mario Veiga Ferraz Pereira

Abstract

Agentic artificial intelligence is emerging as a paradigm in which reasoning-capable models interact with external tools, simulations, and data sources to perform multi-step analytical tasks. For energy-system applications, this raises the question of whether AI agents can operate effectively within structured analytical environments using domain-specific capabilities to explore complex decision problems, rather than merely as wrappers around existing solvers.

This paper investigates this question through a stochastic hydrothermal scheduling case study, which combines uncertainty, intertemporal coupling, physical constraints, and economically meaningful operational trade-offs while offering reliable optimization benchmarks such as stochastic dual dynamic programming (SDDP). We propose a capability-driven agentic architecture in which a reasoning agent interacts with the hydrothermal model only through controlled, typed domain capabilities, without access to pre-programmed dynamic programming methods, precomputed water values, dual variables, or internal solver information.

The architecture is evaluated on a simplified but structurally realistic representation of the Brazilian Interconnected Power System. Across ten independent sessions, the agent inferred water-value logic from simulation feedback, constructed approximate future-cost representations, and deployed storage-preserving policies that substantially improved upon myopic behavior. Relative to an independently computed SDDP benchmark, the best session achieved a 1.7% mean cost gap, while the ten-session average gap was approximately 6.0%.

These results indicate that structured capability orchestration can enable reasoning agents to function as analytical layers embedded in energy-system modeling environments, most valuably for problems where analytical formulations are incomplete, difficult to specify, or insufficient to capture the full solution-exploration process.

Keywords: agentic AI, hydrothermal scheduling, stochastic optimization, water value, SDDP

Introduction

Agentic artificial intelligence is emerging as a paradigm in which reasoning-capable language models interact with external tools, simulations, data sources, and computational environments to perform multi-step analytical tasks. Instead of producing only static text outputs, these systems can select structured actions, invoke tools, interpret returned observations, revise hypotheses, and coordinate multi-step analytical workflows toward a user-defined objective [1]. Recent agentic frameworks commonly separate a language-model inference core, responsible for reasoning, planning, and action selection, from an orchestration layer, or Harness, that manages tool calls, observations, memory, and execution state across steps [4, 2, 3]. This general structure is illustrated in Fig. 1, which shows the LLM inference core and the surrounding Harness.

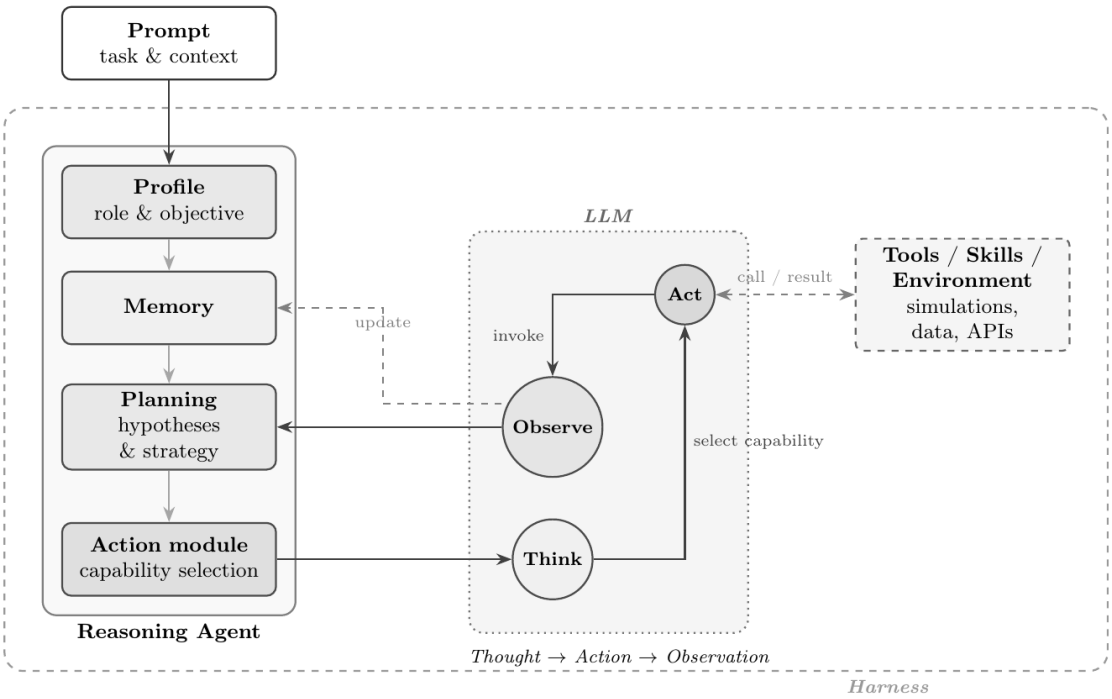


Figure 1: General structure of an LLM-based agentic reasoning loop.

Although these developments have been demonstrated primarily in generic enterprise, software, coding, and information-retrieval applications, they raise a relevant methodological question for energy-system analytics: can reasoning agents operate effectively inside structured analytical environments for complex engineering decision problems, or do they merely provide natural-language interfaces to existing computational tools? This question is particularly important in energy-system planning and operation, where decision-support processes rely on specialized models that represent uncertainty, physical constraints, intertemporal couplings, and economically meaningful trade-offs.

Hydrothermal scheduling provides a suitable setting for investigating this question. In hydrothermal systems, current reservoir-release decisions reduce present thermal generation costs but also affect future exposure to scarcity, expensive thermal dispatch, and energy deficits. Stored water therefore has an opportunity cost, commonly represented through water values or future-cost functions. The problem combines stochastic inflows, storage dynamics, transmission constraints, thermal substitution, and long-term operational trade-offs. At the same time, it has well-established stochastic optimization references, particularly Stochastic Dual Dynamic Programming (SDDP), which provides a rigorous benchmark for evaluating the quality of operating policies [5, 6].

Classical stochastic dynamic programming provides the conceptual basis for this sequential decision problem, but direct application is limited by the curse of dimensionality. In realistic hydrothermal systems, the state vector includes reservoir storage levels, hydrological conditions, and other intertemporal variables, causing the number of possible states to grow exponentially with system size and uncertainty representation. Decomposition methods such as SDDP avoid full state enumeration by approximating future-cost functions and remain the standard reference for large-scale hydrothermal operation planning. Machine learning methods, including reinforcement learning, have also been studied for sequential decision-making under uncertainty. In this paper, however, these methods serve primarily as conceptual references and benchmarks: the objective is not to propose a new hydrothermal optimization solver.

The central question addressed in this paper is instead architectural and methodological: can a reasoning agent, restricted to controlled domain capabilities, explore an energy-system analytical environment and

construct an operationally meaningful policy without access to a pre-programmed stochastic dynamic programming method, dual variables, precomputed water values, or solver internals? This question concerns how analytical functionality should be exposed to an AI agent so that reasoning, computation, control, and auditability remain separated. It is aligned with the emerging notion of domain-specific skills or capabilities: modular operations that expose selected functionality of an environment to an agent through structured interfaces.

To investigate this question, we propose a capability-driven agentic architecture for AI-assisted interaction with energy-system analytical environments. In this architecture, domain knowledge and model operations are exposed to the agent through structured and typed capabilities rather than through unrestricted code execution or direct access to solver internals. The agent can inspect the system, query scenario statistics, run simulations, evaluate candidate policies, and register policy-relevant artifacts, but all computations are executed by the controlled analytical environment. The agent therefore does not replace the domain model or the optimization solver; it acts as an analytical orchestration layer that formulates hypotheses, invokes capabilities, interprets outputs, and progressively constructs an approximate operating logic.

The proposed architecture is instantiated in a stochastic hydrothermal scheduling environment based on a simplified but structurally realistic representation of the Brazilian Interconnected Power System. The agent interacts with the model only through controlled capabilities and is not given access to an SDDP implementation, dynamic programming routines, dual information, precomputed water values, or internal solver state. Its resulting policy is then evaluated through stochastic simulation and compared with two references: a myopic policy that ignores future water value and an independently computed SDDP-based benchmark. This setting allows the agent's behavior to be assessed in a problem where the optimality structure is well understood and where reliable benchmarks are available.

The broader motivation is to evaluate whether reasoning agents can function as controlled analytical layers around energy-system models. Such agents are not expected to replace formal optimization methods. Rather, they may support exploration, interpretation, sensitivity analysis, model interrogation, and policy construction in settings where the relevant decision process extends beyond a single fully specified optimization formulation. Hydrothermal scheduling is used here as a benchmarked validation case before considering more open-ended energy-system problems in which analytical formulations may be incomplete, difficult to construct, or insufficient to capture the full solution-exploration process.

The paper addresses the following research questions:

RQ1. Can a reasoning agent use structured domain capabilities to construct an operationally meaningful policy in a stochastic energy-system scheduling problem?

RQ2. In the hydrothermal scheduling case, can the agent infer economically meaningful water-value logic from simulation feedback rather than from explicit dual information or precomputed optimization outputs?

RQ3. How reproducible is the policy-construction process across independent reasoning-agent sessions under the same task specification and capability interface?

RQ4. What architectural properties are required to support controlled, auditable, and modular evaluation of reasoning agents in energy-system analytical environments?

The main contributions of the paper are:

1. A capability-driven agentic architecture for AI-assisted interaction with energy-system analytical environments.
2. An instantiation of this architecture in a stochastic hydrothermal scheduling environment, connecting the agentic workflow to the classical multistage stochastic optimization formulation.

3. A structured capability interface that exposes controlled domain operations for model inspection, simulation-based exploration, policy registration, and stochastic evaluation while withholding solver internals, dual variables, and precomputed water values.
4. A benchmarked experimental assessment showing that a reasoning agent can construct storage-preserving hydrothermal operating policies through structured capability orchestration, with comparative evaluation against myopic operation and independently computed SDDP-based references.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the multistage stochastic hydrothermal operation problem. Section 4 presents the capability-driven architecture, the analytical environment, and the experimental protocol. Section 5 presents the test case and results. Section 6 discusses implications and limitations. Section 7 concludes.

Related Work

Recent work on tool-using language models has shown that large language models can combine language-mediated reasoning with external computational actions, including tool calls, API access, and structured interaction with computational environments [1, 7, 8]. Related agentic architectures organize this process through modules for role specification, memory, planning, and action selection, and increasingly emphasize reusable skills or capabilities as the interface between the reasoning model and its environment [2, 3]. Industrial frameworks have followed a similar direction by providing reference patterns for combining foundation models, tool access, workflow orchestration, and guardrails [9, 10].

While many demonstrations of tool-using agents focus on web search, coding, question answering, or information retrieval, high-consequence engineering applications require stronger guarantees of control, auditability, and reproducibility. In such settings, unrestricted tool use is often inappropriate: the agent must interact with validated computational environments through well-defined interfaces. Interoperability protocols such as the Model Context Protocol (MCP) illustrate this direction by exposing external tools through typed interfaces [11]. This motivates the use of controlled capability interfaces for agentic interaction with energy-system models.

Machine learning has also been extensively applied in energy systems, including forecasting, optimal power flow approximation, and control [12, 13]. Reinforcement learning has been studied for reservoir operation and power-system control [14, 15]. These approaches usually learn policies or function approximations through data, simulation episodes, gradient updates, or explicit training procedures. The approach investigated here is different: the agent is not trained as a hydrothermal controller and does not learn a policy through repeated episodes. Instead, it uses reasoning and capability orchestration to inspect a structured energy-system environment, test hypotheses, and construct a policy-relevant approximation during an analytical session.

Hydrothermal operation planning has long been addressed through stochastic dynamic programming and decomposition. SDDP [5] represents the future-cost function through affine cuts and is widely used in large-scale hydrothermal systems. Subsequent work extended the method to convergence analysis, risk aversion, and intertemporal inflow models [6, 16, 17]. These methods provide the optimization benchmark for the present work and are used as external references, not as tools available to the agent.

Multistage Stochastic Hydrothermal Operation Problem

Consider a finite horizon $t = 1, \dots, T$. Let V_t be reservoir storage, H_t hydrological information, and $X_t = (V_t, H_t)$ the system state. Uncertainty is represented by ξ_t, including inflows and possibly demand or renewable generation. At each stage, decisions u_t include hydro generation, thermal generation, interconnection flows, spillage, deficit and next-stage storage. The stage problem is

$$\min_{u_t \in U_t(X_t, \xi_t)} c_t(u_t, \xi_t) + Q_{t+1}(X_{t+1}), \qquad (1)$$

where U_t contains water balance, generation, transmission and demand constraints.

For each hydro subsystem r, storage evolves as

$$V_{t+1,r} = V_{t,r} + a_{t,r}(\xi_t) - q_{t,r} - s_{t,r}, \qquad g^H_{t,r} = \eta_{t,r}\, q_{t,r}, \qquad (2)$$

with inflow $a_{t,r}$, turbined outflow $q_{t,r}$, spillage $s_{t,r}$ and production coefficient $\eta_{t,r}$. For each electrical area n,

$$\sum_{r \in R_n} g^H_{t,r} + \sum_{g \in G_n} g^T_{t,g} + R_{t,n}(\xi_t) + \sum_{\ell \in I_n} f_{t,\ell} - \sum_{\ell \in O_n} f_{t,\ell} + d_{t,n} = D_{t,n}(\xi_t). \qquad (3)$$

The feasible set also includes bounds on storage, hydro generation, thermal generation, transmission flows, spillage and deficits. The immediate cost is

$$c_t(u_t, \xi_t) = \sum_n \sum_{g \in G_n} C^T_{t,g}\; g^T_{t,g} + \sum_n C^D_{t,n}\; d_{t,n}. \qquad (4)$$

The multistage stochastic problem is

$$\min_{\pi} \mathbb{E}\left[\sum_{t=1}^{T} c_t\,(u_t, \xi_t) + \Phi(V_{T+1})\right], \qquad u_t = \pi_t(X_t, \xi_t), \qquad (5)$$

with nonanticipative policies. The Bellman recursion is

$$Q_t(X_t) = \mathbb{E}_{\xi_t | X_t}\left[\min_{u_t \in U_t(X_t, \xi_t)} \{c_t(u_t, \xi_t) + Q_{t+1}(X_{t+1})\}\right], \qquad (6)$$

with terminal condition $Q_{T+1}(X_{T+1}) = \Phi(V_{T+1})$. The marginal value of stored water is

$$\lambda_{t,r}(X_t) = \frac{\partial Q_t(X_t)}{\partial V_{t,r}}. \qquad (7)$$

Direct SDP discretization is intractable because R reservoirs discretized into K levels already produce K^R storage states. SDDP avoids full state enumeration by approximating future costs with affine cuts:

$$Q_t(V) \approx \max_{k \in K_t} \{\alpha^k_t + (\beta^k_t)^\top V\}. \qquad (8)$$

In this paper, SDDP is used as conceptual reference and external benchmark. The agent is not given access to an SDDP implementation.

Capability-Driven Architecture

Architectural principle

The proposed capability-driven architecture (CDA) separates the agent's reasoning process from the computational implementation of the underlying energy-system analytical model. The agent formulates hypotheses, selects analytical actions, interprets observations, and refines its strategy over a multi-step session, while the environment executes validated domain operations such as model inspection, simulation, policy evaluation, and artifact registration.

All interactions occur through typed capabilities with defined input schemas, output schemas, and execution semantics. The agent has no access to model files, solver internals, unrestricted code execution, dual variables, or precomputed optimization outputs. This separation allows the agent to operate as a reasoning and orchestration layer while preserving control, auditability, and a clear boundary between language-mediated inference and validated computation.

Formally, an analytical session under the CDA can be represented as an interaction trajectory

$$\tau = [(a_1, o_1), (a_2, o_2), \dots, (a_K, o_K)], \qquad (9)$$

where a_k denotes the capability invocation at step k and o_k the corresponding structured observation returned by the environment.

At each step, the agent selects the next capability invocation based on the full interaction history accumulated so far. Let $h_k = (\mathcal{T}, a_1, o_1, \dots, a_{k-1}, o_{k-1})$ denote that history, where $\mathcal{T}$ is the task specification provided at session initialization. The agent's reasoning loop can then be written as

$$a_k = \pi_A(h_k), \qquad (10)$$

where π_A is the agent's orchestration function: the mechanism by which the reasoning model interprets the accumulated history and selects the next capability invocation. This object should not be confused with an operational hydrothermal policy. It governs the sequencing of analytical actions during the agentic session, including inspection, simulation, hypothesis testing, and possible modification of policy-related artifacts such as future-cost approximations. Unlike a trained reinforcement learning controller, π_A is not learned through episodic interaction with the hydrothermal system; it is executed by the reasoning model through language-mediated inference over the session history.

The environment, in turn, maintains a state s_k comprising the loaded analytical model, the available capability definitions, and any stateful analytical artifacts created during the session. In the hydrothermal instantiation, the most important such artifact is the currently registered future-cost approximation.

Each capability invocation produces an observation and advances the environment state according to

$$(o_k, s_{k+1}) = \mathcal{E}(a_k, s_k), \qquad (11)$$

where $\mathcal{E}$ is a deterministic transition function. Read-only invocations (inspection, simulation, and scenario-statistics queries) leave s_k unchanged. Policy-encoding invocations update the registered cut set in s_{k+1}, making the environment stateful with respect to the policy construction process. The session terminates at step K when the agent declares the analytical task complete. The terminal environment state s_K contains the artifacts produced during the session. In the hydrothermal case, these artifacts include the registered future-cost approximation. This approximation subsequently induces an operational hydrothermal policy when embedded in the stage-dispatch problem and evaluated through stochastic simulation, as described in Section 5.

The trajectory τ is the primary object of post-hoc analysis: it records the full analytical path from system inspection to policy validation and enables reconstruction and auditability of the agent's reasoning process.

Figure 2 illustrates the overall component structure of the architecture, summarising the formal relationships described by equations (9)–(11).

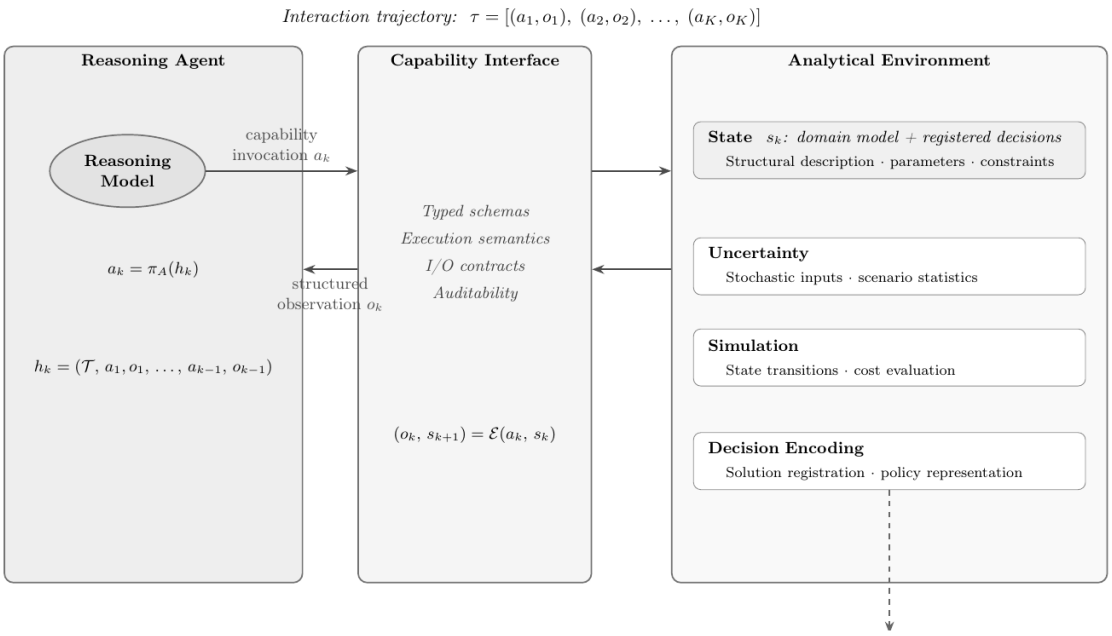


Figure 2: Capability-driven architecture.

Hydrothermal instantiation of the capability interface

The hydrothermal instantiation exposes domain operations to the agent through eight typed capabilities, organized into four analytical groups. The design follows a principle of deliberate information scoping: each capability provides the information required to support a well-defined analytical step while withholding solver internals, algorithmic templates, dual variables, and raw data streams that would allow the agent to bypass the reasoning task. This scoping is not merely a technical convenience; it is the mechanism by which the environment constrains the agent to perform domain-relevant reasoning rather than arbitrary computation or solver wrapping.

A further architectural property is session statefulness. The model is loaded once at session initialization, fixing the reservoir and establishing the storage-vector coordinate convention used in all subsequent simulation and policy-encoding calls. This explicit data contract between the environment and the agent ensures that every capability invocation operates on a shared, unambiguous representation of the physical system.

Table 1 summarizes the practical interface.

Table 1: Hydrothermal capability interface: capability groups, inputs, outputs, and analytical roles.

Group	Key inputs / outputs	Analytical role
System inspection	case data → reservoirs, stages, scenarios	Initialize session; fix storage-vector coordinate convention
	$\emptyset \rightarrow$ full topology	Expose subsystem graph, cost blocks, transmission, demand
	$\emptyset \rightarrow T,\ \lvert S \rvert$	Provide planning horizon and scenario count

Hydrological scenarios	$\emptyset \rightarrow$ mean, std per node per stage	Characterize distributional inflow structure; individual paths not accessible
Operation simulation	stage, scenario, $V \rightarrow$ costs, V'	Single-stage controlled oracle; primal outcomes only
	scenario $\rightarrow$ cost and storage trajectories	Full-horizon trajectory for one inflow path
	$\emptyset \rightarrow$ aggregate results	All-scenario batch evaluation
Policy encoding	stage, β, $\Lambda \rightarrow$ confirmation	Register affine future-cost approximations; replaces existing cuts at that stage

System-inspection capabilities expose the static structural description of the hydrothermal model. A data-loading call initializes the session and returns the ordered reservoir name, initial volume, maximum capacity, and physical unit for each storage component. This ordering is consequential: it defines the coordinate system for all storage vectors exchanged in subsequent calls, from single-stage simulation inputs to policy-cut coefficient matrices. A topology-inspection call provides the complete system description as a structured document, including subsystem nodes, directional interconnection arcs with capacity limits, thermal generation blocks with cost tiers, renewable generation profiles, and seasonal demand parameters. The resulting picture matches what a planning analyst would assemble by reviewing model documentation: system topology, resource characterization, and operating constraints.

Hydrological-scenario capabilities provide aggregate stochastic characterization of input uncertainty. For each input node, including reservoir inflows, renewable generation, and demand perturbations, the capability returns per-stage summary statistics across the scenario ensemble. Individual scenario realizations are intentionally not accessible through this capability. This choice serves two purposes. Analytically, it encourages the agent to reason from distributional structure rather than memorize individual sample paths. Operationally, it avoids injecting large raw scenario tables into the interaction history, which would expand the agent's context with low-level numerical detail and dilute the higher-level analytical signal needed for subsequent reasoning steps. The full scenario ensemble remains available indirectly through simulation and batch evaluation capabilities.

Operation-simulation capabilities constitute the computational core of the interface and offer three levels of analytical granularity. A single-stage simulation call evaluates one stage given an initial storage vector, a stage index, and a scenario index, returning a vector of per-node operating costs and the resulting end-of-stage storage vector. This oracle supports targeted perturbation experiments: by varying the initial storage vector across repeated calls while holding stage and scenario fixed, the agent can estimate the local sensitivity of operating cost to reservoir levels—the numerical procedure that yielded the marginal water values reported in Section 5. A sequence-simulation call runs a complete trajectory for a specified inflow scenario, returning stage-by-stage cost and storage profiles over the full planning horizon. A batch-evaluation call executes the model across all scenarios and is appropriate for final policy validation.

A defining property of this group is that operating costs and state transitions are observable, but dual variables and marginal prices are not. The stage solver executes internally for each simulation call and returns primal outcomes only; shadow prices are not surfaced through the interface. This constraint has direct consequences for the agent's analytical strategy. The opportunity cost of stored water must be inferred through active experimentation rather than read from solver output.

Policy-encoding capabilities allow the agent to register future-cost approximations that are incorporated into subsequent simulation calls. A policy-registration call accepts a stage index, a vector of

right-hand-side constants, and a matrix of storage coefficients, each row defining one affine future-cost approximations on the future cost function. This call replaces all existing cuts for the target stage: incremental accumulation across iteration steps is the agent's responsibility, not the environment's. The storage coefficients follow a sign convention aligned with the hydrothermal economics: a negative coefficient on reservoir i encodes the observation that additional water in that reservoir reduces expected future operating costs, i.e., that stored water has positive marginal value. Conversely, passing empty coefficient and right-hand-side arrays clears the cuts for a stage, enabling the agent to revise or reset its policy approximation during the analytical session.

This taxonomy is specific to the hydrothermal application, but it reflects a broader architectural principle: capabilities should correspond to meaningful analytical operations in the target domain, grouped at a level of abstraction that matches the reasoning granularity required for the task. In this case, the exposed capabilities mirror the workflow followed by hydrothermal planning analysts: inspect the system, characterize the uncertainty structure, simulate operational consequences under controlled conditions, and encode intertemporal policy logic. The deliberate withholding of dual information and individual scenario paths ensures that the agent must engage with the analytical problem rather than retrieve precomputed answers through the interface.

The policy-encoding capability accepts affine future-cost approximations of the form:

$$Q_t(V) \geq \beta_t - \sum_{i \in S} \lambda_i V_i, \qquad (12)$$

where $Q_t(V)$ is the approximation of future cost from stage $t+1$ onward, V_i is end-of-stage storage in subsystem i, λ_i is the marginal value of stored water, β_t is a stage-specific intercept, and S is the set of storage components represented in the model.

This representation is closely related to the affine cuts used in decomposition methods such as SDDP. However, the agent was not provided with an implementation of SDDP, any other pre-programmed stochastic dynamic programming method, or an algorithmic template for constructing these cuts. The capability only defined the admissible policy object that could be registered in the environment. The reasoning process used to estimate water values, choose coefficients, and assemble the approximation was left to the agent.

Results

Experimental task and evaluation

The agent was instructed to construct a cost-effective operating policy for the hydrothermal system. The session was allowed to proceed until the agent declared that it had completed policy construction and validation. The interaction log was then analyzed post hoc. Throughout the session, the agent had access to capability descriptions, input-output schemas, and the outputs of its own invocations. It did not receive an implementation of dynamic programming, such as SDDP, precomputed water values or dual variables, access to internal solver state, or human guidance after task initialization. All quantitative information used by the agent was therefore obtained exclusively through structured capability invocations, distinguishing this experimental setting from one in which an agent simply calls an existing optimization solver.

All sessions were conducted using Claude Sonnet 4.6 (Anthropic), a large language model with extended chain-of-thought reasoning capabilities. The model was deployed with extended thinking enabled, which allows the model to perform explicit intermediate reasoning steps before producing each response or capability invocation. Sessions used an automatic orchestration mode in which the primary reasoning model autonomously routes computationally lighter sub-tasks, such as structured-output parsing and

invocation formatting, to Claude Haiku 4.5, a smaller and lower-latency member of the same model family, while retaining full extended-thinking inference for analytical steps that require multi-step reasoning. This configuration reflects a practical deployment pattern for long-horizon agentic workflows, in which reasoning depth and computational efficiency are balanced automatically at the inference level without manual intervention.

The experiment was designed to evaluate whether the agent could:

1. Identify relevant hydrothermal system structure through inspection capabilities;
2. Infer intertemporal water-value logic through simulation;
3. Encode a future-cost approximation using the available policy representation;
4. Validate the resulting policy across stochastic inflow scenarios.

The resulting policy was evaluated using sample-mean operating cost, scenario cost variability, energy-deficit events, use of high-cost emergency thermal generation, reservoir storage trajectories, and scenario-level comparison with two references:

- **Myopic baseline:** a short-sighted policy that prioritizes immediate hydro generation without accounting for future water value, corresponding to solving each stage independently with a zero future-cost approximation.
- **SDDP benchmark:** a policy obtained independently using a purpose-built stochastic dual dynamic programming implementation.

Because reasoning agents are stochastic systems, in which their outputs depend on internal sampling processes that vary across sessions even under identical prompts and initial conditions, the experiment was conducted ten times independently, each run initialized from the same model and task specification. The ten runs produce different interaction trajectories $\tau^{(r)}$, $r = 1, \ldots, 10$, potentially leading to different policy objects $\pi_{\tau}^{(r)}$ and different operational costs. Analyzing the distribution of outcomes across runs prevents conclusions from being drawn based on a single, potentially atypical session.

Brazilian hydrothermal test case

The experiment used a simplified but structurally realistic representation of the Brazilian Interconnected Power System (SIN). The system is represented by four interconnected subsystems: Southeast/Centre-West (SECO), South (SUL), Northeast (NE), and North (N). The planning horizon covers 24 monthly stages, from January 2025 to December 2026, with ten stochastic inflow scenarios.

The test case includes aggregate hydro reservoirs, layered thermal generation blocks, renewable generation, demand profiles, and inter-regional transmission links. It is not intended to reproduce the full official Brazilian operation model. Rather, it provides a controlled benchmark with the key structural characteristics required for hydrothermal operation analysis: spatial coupling, storage dynamics, hydrological uncertainty, thermal substitution, and scarcity penalties.

Table 2 summarizes the main case parameters.

Parameter	**Value**
Planning horizon	January 2025–December 2026
Stages	24 monthly stages
Subsystems	SECO, SUL, NE, N
Hydrological scenarios	10 stochastic inflow scenarios

Parameter	Value
Policy representation	Affine future-cost cuts

Table 2: Main test-case parameters.

Table 3 summarizes the main subsystem characteristics used in the test case.

Subsystem	Hydro max. (MW)	Reservoir (GWh)	Initial storage	Demand range (MW)
SECO	35,000	146,000	75%	42,800–50,000
SUL	17,000	14,600	70%	12,900–15,800
NE	10,000	37,960	50%	12,400–14,000
N	22,000	10,950	85%	7,400–8,400

Table 3: Aggregate subsystem characteristics in the hydrothermal test case.

The test case also includes transmission links among the subsystems. The main directional limits are 15,600 MW from N to SECO, 2,000 MW from SECO to N, 6,200 MW from N to NE, 13,000 MW from NE to SECO, and 8,000 MW between SECO and SUL. Thermal generation is represented by layered blocks with increasing costs. The SECO subsystem plays a central role because its high-cost thermal block at 600 $/MWh frequently acts as a system-wide marginal resource in scarcity conditions.

Figure 3 shows the schematic of the test case, illustrating the four subsystems and the directional transmission links connecting them.

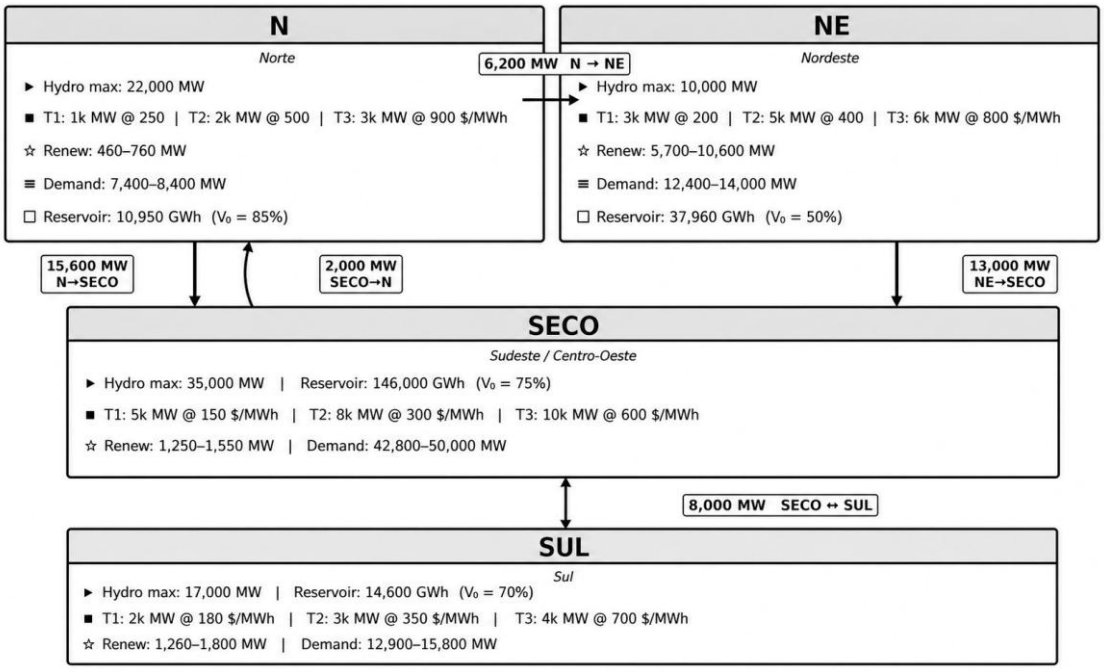


Figure 3: Schematic of the hydrothermal test case based on the Brazilian Interconnected System (SIN).

Hydrological scenario ensemble

The ten inflow scenarios were constructed to represent structurally distinct hydrological regimes rather than random perturbations around a central estimate. Table 4 reports each scenario's type and annual inflow deviation from the ensemble mean per subsystem and for the system as a whole.

Scenario	Type	SECO	SUL	NE	N	System
1	Normal	+10	−4	+4	0	+3
2	Normal	0	+11	+4	+2	+3
3	Normal	+14	+2	+10	+9	+9
4	Wet	+31	+34	+30	+31	+31
5	Wet (La Niña)	+45	0	+36	+39	+35
6	Wet	+25	+33	+21	+29	+28
7	Dry	−25	−20	−19	−25	−24
8	Dry (El Niño)	−38	+15	−28	−23	−23
9	Dry	−14	−28	−12	−15	−16
10	Critical	−47	−44	−45	−47	−46

Table 4: Hydrological scenario ensemble: type and annual inflow deviation from ensemble mean (%).

The ensemble spans three near-average realizations (scenarios 1–3), three wet years (4–6, +28% to +35% above mean), three dry years (7–9, −16% to −24% below mean), and one critical scenario (10) modelled after the 2001 Brazilian energy crisis, with all subsystems approximately 47% below their mean inflows. Total annual system inflows range from 233 300 GWh to 586 900 GWh, a factor of 2.5×, with individual subsystem ratios reaching 2.7× for SECO.

Beyond aggregate severity, several scenarios present pronounced cross-subsystem heterogeneity. Scenario 5 reproduces a La Niña-type pattern in which SECO, NE, and N are well above average (+45%, +36%, +39%) while SUL receives near-average inflows. Scenario 8 inverts this in an El Niño configuration: SECO, NE, and N severely below average (−38%, −28%, −23%) while SUL is above average (+15%). Together, the ten scenarios cover a broad spectrum of aggregate severity and spatial inflow structure, providing a demanding test of policy robustness despite the modest ensemble size.

Marginal water-value estimation

As established in the experimental design, each of the ten runs followed an independent reasoning trajectory. Despite this, inspection of the interaction logs revealed a consistent pattern across runs: the agent recurrently converged on a finite-difference approach to estimate the marginal value of stored water, independently of the specific sequence of tool invocations that preceded it. This convergence suggests that the approach is a natural consequence of the available capabilities and the problem structure, rather than an artefact of a particular session.

For each subsystem i, it compared the simulated operating cost at a reference storage vector with the cost obtained after perturbing that subsystem's storage by a small increment ΔV:

$$\hat{\lambda}_i = -\frac{C(V + \Delta V e_i) - C(V)}{\Delta V}. \qquad (13)$$

Backward construction of future-cost cuts

A second pattern that recurred consistently across runs was the strategy for constructing future-cost cuts. After estimating per-subsystem water values through finite differences, the agent systematically proceeded to encode them as affine future-cost approximations on the cost-to-go function by backward

induction — a procedure that emerged independently in each session without being prescribed in the task specification.

Using the per-subsystem water-value coefficients $\hat{\lambda}_i$ estimated in the previous step, the agent constructed a sequence of future-cost cuts by backward induction. Starting with a zero terminal value, for each stage $t = T, \ldots, 1$, it simulated operation from an initial state and computed:

$$\beta_t = c_t - \sum_i \hat{\lambda}_i V_{t,i}^{\text{fin}} + \beta_{t+1}, \qquad (14)$$

where c_t is the immediate operating cost returned by the simulator and $V_{t,i}^{\text{fin}}$ is the resulting end-of-stage storage in subsystem i.

The resulting policy is represented by an affine future-cost approximations on future cost:

$$Q_t(V) \geq \beta_t - \sum_i \hat{\lambda}_i V_i. \qquad (15)$$

Algorithm 1 summarizes the procedure reconstructed from the interaction log.

Algorithm 1 Agent-derived backward construction of affine future-cost cuts

Require: Per-subsystem water-value coefficients $\hat{\lambda}_i$, terminal intercept $\beta_{T+1} = 0$

1. **for** $t = T, T-1, \ldots, 1$ **do**
2. Simulate stage t from zero initial storage
3. Observe immediate cost c_t and final storage vector V_t^{fin}
4. Compute $\beta_t = c_t - \sum_i \hat{\lambda}_i V_{t,i}^{\text{fin}} + \beta_{t+1}$
5. Define cut $Q_t(V) \geq \beta_t - \sum_i \hat{\lambda}_i V_i$
6. **end for**
7. Register all non-terminal cuts with the optimization environment

This procedure resembles a simplified single backward pass of SDDP, but it was not supplied to the agent as an algorithmic template. The intercept profile decreases as the horizon approaches its terminal stage, reflecting the declining value of future operating periods.

Multi-run reproducibility analysis

Table 5 reports the mean operating cost obtained in each of the ten independent runs, listed in session order.

Run	Mean cost (M$)
R1	139.24
R2	148.33
R3	145.77
R4	153.26
R5	138.12
R6	153.94
R7	139.24

Run	Mean cost (M$)
R8	139.97
R9	139.21
R10	137.72
Grand mean	143.48
Std dev	6.02
Min	137.72
Max	153.94

Table 5: Mean total operating cost for each independent run (M$).

Six runs (R1, R5, R7–R10) produced policies with mean costs in the range 137.7–140.0 M$, a spread of only 2.3 M$ across six independent sessions. The remaining four runs (R2–R4, R6) yielded higher mean costs of 145.8–153.9 M$. The grand mean across all ten runs is 143.5 M$ with a standard deviation of 6.0 M$. A noteworthy observation is that runs R1 and R7 produced identical per-scenario costs to within numerical precision, indicating that the agent arrived at exactly the same policy through two independent interaction trajectories—a form of convergence that suggests the analytical path to the dominant water-value estimate is sufficiently constrained by the capability structure to be reproducible.

Reasoning trajectory of the best-performing run

Among the ten independent runs, the session that achieved the lowest mean operating cost (137.7 M$) left a sufficiently detailed interaction log to reconstruct its analytical trajectory in full. The agent began with system inspection (Phase 1), loading the model data, cost structure, and scenario statistics. In Phase 2, it simulated all ten scenarios without any registered cuts, establishing a baseline in which reservoirs drained between stages 6 and 8 in every scenario and the system operated entirely on thermal dispatch from stage 8 onward—the classic myopic collapse.

Phase 3 consisted of targeted single-stage perturbation experiments. By perturbing each reservoir in isolation at a stressed stage and scenario, the agent estimated the marginal value of stored water: approximately 600 $/MWh for SECO, SUL, and N, and 567 $/MWh for NE. A key inference was that all four subsystems were dominated by the same system-wide marginal resource—the third thermal block in SECO at 600 $/MWh—reflecting the high degree of interconnection in the system.

In Phase 4, the agent registered a first set of affine cuts with coefficients set slightly more negative than the estimated threshold ($\alpha = [-700, -700, -600, -700]$ $/MWh). Evaluating this policy revealed an inconsistency: SECO storage was accumulating while the expensive thermal block was simultaneously being dispatched, indicating that the coefficients were penalising water use too aggressively. The agent identified this over-conservation and revised the coefficients downward in Phase 5 ($\alpha_{\mathrm{SECO}} = -590$, $\alpha_{\mathrm{NE}} = -560$), aligning them with the marginal values estimated in Phase 3. The refined policy produced a mean operating cost of approximately 138 M$ and was accepted as the final output.

This trajectory is notable for its self-correcting structure. Without external guidance, the agent detected a policy inconsistency through simulation feedback and revised its representation of water value accordingly.

Comparison with myopic operation

Table 6 compares the agent-derived policy with the myopic baseline.

Indicator	Myopic	Agent
Mean cost (M$)	275.7	137.7
T3 emergency (stages)	10	0
Deficit events (stages)	4	0
Mean savings (M$)	–	138.0

Table 6: Comparison between myopic operation and the agent-derived policy.

The myopic policy rapidly depletes reservoirs during the first wet season, leaving insufficient hydraulic buffer for later dry periods. The agent-derived policy instead preserves storage, accepting higher short-term thermal costs to reduce future scarcity risk. This behavior is consistent with the economic interpretation of water values in stochastic hydrothermal operation: water should be used when its immediate benefit exceeds its expected future opportunity cost.

Figure 4 presents the stochastic evolution of reservoir storage across the ten inflow scenarios under the agent-derived policy and the myopic baseline. Each panel corresponds to one subsystem. Bold lines show the scenario mean; shaded bands report the P25–P75 and P10–P90 intervals across scenarios. Storage levels are expressed as a percentage of each subsystem's maximum capacity.

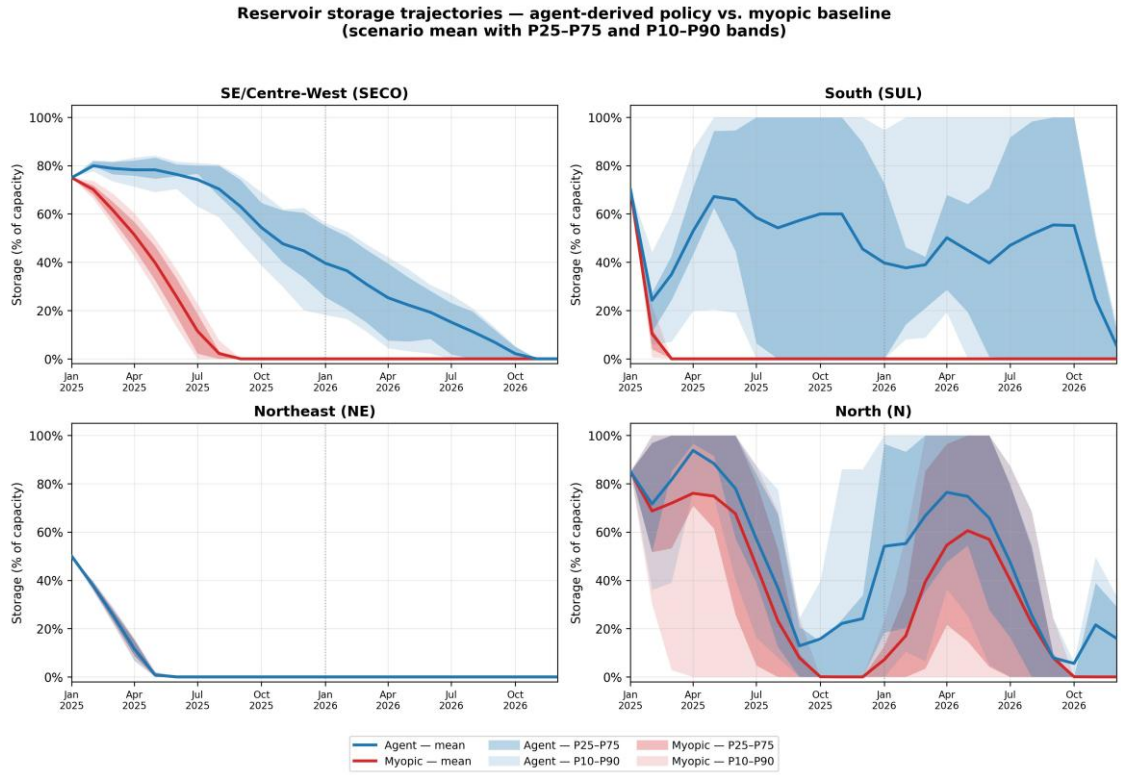


Figure 4: Reservoir storage trajectories under the agent-derived policy and the myopic baseline. Red curves denote the myopic baseline.

The contrast between the two policies is pronounced. Under the myopic baseline, all subsystems deplete to near-zero levels by stages 7–8 (July–August 2025) in most scenarios, and the system operates entirely on thermal dispatch for the remainder of both dry seasons. Under the agent-derived policy, the SECO reservoir, which concentrates approximately 70% of total system storage capacity, is maintained well above zero throughout the first dry season and enters 2026 with sufficient hydraulic reserve to buffer

the second dry period. The NE and N subsystems, which have smaller absolute capacities, exhibit similar qualitative differences, while SUL is partly recovered by its own inflow regime.

Comparison with SDDP

Table 7 compares the best-performing agent run (R10, mean cost 137.7 M$) with the SDDP benchmark using matched inflow realizations. This run is selected here because it produced the lowest mean cost among the ten independent sessions and thus represents the most favourable outcome of the capability-driven approach. Results for the remaining runs are reported in Section 5.5; across all ten runs, the mean cost gap relative to SDDP is approximately 6.0%, reflecting the variability introduced by the probabilistic nature of the reasoning agent. The SDDP reference was obtained using the PSR commercial solver [18], applied to the same ten inflow scenarios with five backward scenarios per iteration.

Scenario	Agent (M$)	SDDP (M$)	Diff. (M$)	Gap (%)
1	113.5	114.5	−1.0	−0.9
2	111.6	107.2	4.4	4.1
3	90.0	81.1	8.9	11.0
4	70.0	64.0	6.0	9.4
5	73.7	66.3	7.4	11.2
6	128.8	126.3	2.5	2.0
7	193.4	191.8	1.6	0.8
8	178.7	178.5	0.2	0.1
9	231.9	237.9	−6.0	−2.5
10	185.7	186.6	−0.9	−0.5
Mean	137.7	135.4	2.3	1.7
Std. dev.	53.2	58.5	–	–

Table 7: Scenario-level comparison between the best-performing agent run (R10) and the SDDP benchmark.

For the best run, the mean cost gap relative to SDDP is 1.7%. This result should be interpreted cautiously. The SDDP policy is obtained using a specialized stochastic optimization algorithm with repeated forward-backward passes, whereas the agent-derived policy is a simplified affine policy constructed through structured interaction. The relevant observation is therefore not that the agent competes with SDDP, but that the capability-driven environment enabled the agent to construct a policy with coherent intertemporal water-value logic.

In three scenarios, the agent-derived policy achieved lower ex post cost than the SDDP benchmark. This does not imply dominance over SDDP. Rather, it reflects scenario-level variation and the conservative nature of the agent-derived policy, which tends to preserve storage more aggressively. On average, the SDDP policy remains the better stochastic optimization benchmark.

Discussion

Structured capability orchestration

The experiment shows that a reasoning agent can use structured capabilities to carry out domain-relevant analytical operations in a stochastic energy-system environment. Its interaction trajectory resembled the workflow of a human analyst working with a simulation model: the agent did not receive an optimal policy directly, but progressively developed an approximate operating logic that guided subsequent decisions.

The important point is not that the agent discovered a new optimization method. Rather, the capability structure constrained the interaction in a productive way. By exposing capabilities at appropriate levels of abstraction, the environment enabled the agent to formulate and test operational hypotheses, infer intertemporal water-value logic, and construct an approximate representation of the cost-to-go structure underlying the stochastic dynamic programming problem.

Implications for AI-assisted energy analytics

The proposed architecture suggests a role for reasoning agents as analytical intermediaries embedded in energy-system modeling environments. Rather than replacing simulators, optimization solvers, or human experts, agents can support the exploratory layer around formal models by selecting computations, interpreting outputs, testing hypotheses, and coordinating iterative analysis.

This role is especially relevant for energy problems whose decision context is broader than a single well-specified optimization formulation. Planning and operational studies often combine uncertain assumptions, heterogeneous data, multiple models, regulatory constraints, market-design considerations, and trade-offs that are difficult to encode completely in closed analytical form. In such cases, an agentic system may be useful not because it directly computes an optimum, but because it helps structure the exploration of alternatives, sensitivities, scenarios, and policy-relevant insights.

Potential applications include:

- Exploration of large stochastic or stress-test scenario spaces;
- Comparative analysis across alternative assumptions and model configurations;
- Identification of sensitivities, bottlenecks, and operational drivers;
- Iterative refinement of candidate policies or planning alternatives;
- Orchestration of multiple simulation, forecasting, and optimization tools;
- Explanation and interrogation of model results for decision support.

In this view, capability design becomes central. The agent can only reason effectively about a domain model if relevant analytical operations are exposed in a controlled, semantically meaningful, and auditable way. Hydrothermal scheduling provides a benchmarked validation setting for this architecture because reliable optimization references are available. Future work should evaluate whether similar agentic frameworks can support solution exploration in problems where fully specified analytical formulations are incomplete, difficult to construct, or insufficient to represent the relevant decision context.

Limitations

Several limitations must be acknowledged. Some are specific to the hydrothermal scheduling case study used for validation, while others concern the broader use of reasoning agents in structured energy-system analytical environments.

Regarding the particular hydrothermal scheduling case study, the experiment was conducted on a simplified representation of the Brazilian SIN. Although structurally realistic, it does not capture the full complexity of real-world operation, including detailed unit commitment, security constraints, individual plant constraints, market rules, regulatory restrictions, and operational reserve requirements. The case should therefore be interpreted as a controlled benchmark for evaluating the architecture, not as a full representation of production-grade hydrothermal operation.

Second, the agent-induced procedure has no optimality guarantees. The quality of the resulting operating behavior depends on the hypotheses formulated by the agent, the sequence of capability invocations, and the capabilities made available by the environment. Classical stochastic optimization methods systematically explore the mathematical formulation and provide convergence guarantees under appropriate assumptions; the agent-derived procedure does not.

Finally, the scalability of sequential capability orchestration remains an open issue. Larger systems may require batch simulation capabilities, parallel scenario evaluation, stronger policy templates, context-management mechanisms, and tighter integration with production-grade solvers.

Conclusions

This paper evaluated whether a reasoning agent can operate as an analytical layer within a structured energy-system modeling environment. To this end, we proposed a capability-driven framework in which the agent interacts with the analytical environment only through structured, typed capabilities. This design separates reasoning and orchestration from validated domain computation, allowing the agent to inspect the model, run simulations, test hypotheses, and register policy-relevant artifacts without unrestricted code execution or access to solver internals.

The framework was assessed on a stochastic hydrothermal scheduling test case based on a four-subsystem representation of the Brazilian SIN. Without access to a pre-programmed stochastic dynamic programming method, dual variables, precomputed water values, or internal solver information, the agent inferred water-value logic from simulation feedback and constructed affine future-cost approximations. When embedded in the stage-dispatch model, these approximations induced storage-preserving operating policies that substantially improved upon myopic operation. Across ten independent sessions, the best policy achieved a 1.7% mean cost gap relative to an independently computed SDDP benchmark, while the average gap across sessions was approximately 6.0%.

These results do not suggest that reasoning agents should replace stochastic optimization methods. Rather, they show that, when constrained by an appropriate capability interface, agentic AI can support the exploratory layer around energy-system models: formulating hypotheses, selecting computations, interpreting outputs, and constructing policy-relevant artifacts while simulation and optimization tools remain the computational foundation.

Future work should evaluate similar capability-driven frameworks in energy-system problems where the relevant decision process extends beyond a single fully specified optimization formulation, including settings that require scenario exploration, coordination of multiple models, sensitivity analysis, and iterative policy assessment.

References

[1] S. Yao, J. Zhao, D. Yu, et al., ReAct: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations, 2023.

[2] L. Wang, C. Ma, X. Feng, et al., A survey on large language model based autonomous agents, Frontiers of Computer Science 18 (2024) 186345.

[3] T. Masterman, S. Besen, M. Sawtell, A. Chandra, The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: a survey, arXiv preprint arXiv:2404.11584 (2024).

[4] T.R. Sumers, S. Yao, K. Narasimhan, T.L. Griffiths, Cognitive architectures for language agents, Transactions on Machine Learning Research (2024).

[5] M.V.F. Pereira, L.M.V.G. Pinto, Multi-stage stochastic optimization applied to energy planning, Mathematical Programming 52 (1991) 359–375.

[6] A. Shapiro, W. Tekaya, J.P. da Costa, M.P. Soares, Risk neutral and risk averse stochastic dual dynamic programming method, European Journal of Operational Research 224 (2013) 375–391.

[7] T. Schick, J. Dwivedi-Yu, R. Dessì, et al., Toolformer: Language models can teach themselves to use tools, Advances in Neural Information Processing Systems 36 (2023).

[8] Y. Qin, S. Liang, Y. Ye, et al., ToolLLM: Facilitating large language models to master real-world APIs, in: International Conference on Learning Representations, 2024.

[9] NVIDIA, NVIDIA AI Blueprints: reference workflows for building generative AI and agentic applications, Technical Documentation, 2024. URL: https://www.nvidia.com/en-us/ai/blueprints/.

[10] NVIDIA, NeMo Guardrails: programmable guardrails for conversational AI applications, Technical Documentation, 2024. URL: https://docs.nvidia.com/nemo/guardrails/.

[11] Anthropic, Model Context Protocol specification, Technical Documentation, 2024. URL: https://modelcontextprotocol.io.

[12] S. Haben, S. Arora, G. Giasemidis, M. Voss, D. Vukadinović Greetham, Review of low voltage load forecasting: methods, applications, and recommendations, Applied Energy 304 (2021) 117798.

[13] A. Venzke, G. Qu, S. Low, S. Chatzivasileiadis, Learning optimal power flow: worst-case guarantees for neural networks, in: Proc. IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, IEEE, 2020.

[14] A. Castelletti, S. Galelli, M. Restelli, R. Soncini-Sessa, Tree-based reinforcement learning for optimal water reservoir operation, Water Resources Research 46 (2010) W09507.

[15] T. Yang, L. Zhao, W. Li, A.Y. Zomaya, Reinforcement learning in sustainable energy and electric systems: a survey, Annual Reviews in Control 49 (2020) 145–163.

[16] A.B. Philpott, Z. Guan, On the convergence of stochastic dual dynamic programming and related methods, Operations Research Letters 36 (2008) 450–455.

[17] P. Girardeau, V. Leclère, A.B. Philpott, On the convergence of decomposition methods for multistage stochastic convex programs, Mathematics of Operations Research 40 (2015) 130–145.

[18] PSR Energy Consulting and Analytics, SDDP – Stochastic Dual Dynamic Programming software, Software Documentation, 2024. URL: https://www.psr-inc.com/pt-br/software/sddp/.

inDepth

GPU-BASED ALGORITHMS FOR LARGE-SCALE OPTIMIZATION

Rafael Benchimol Klausner, Raphael Sampaio, Joaquim Dias Garcia

Introduction

Many of the analytical problems that define modern power systems are, at their computational core, large optimization problems. Economic dispatch, unit-commitment, expansion planning, stochastic operation, reliability assessment, and decomposition-based planning workflows all rely heavily on mathematical programming. As systems incorporate more renewable variability, storage technologies, network detail, and scenario-based representations of uncertainty, the size of these optimization problems is growing faster than traditional workflows were designed to absorb.

This is the context for the technical partnership between PSR and NVIDIA: evaluating how GPU-based algorithms can be used to solve large-scale optimization problems that arise in real power-system analytics. The focus is not simply to run an existing model on a faster device. It is to understand which classes of optimization algorithms benefit from GPU architecture, where the practical bottlenecks move, and how this changes the scale of problems that can be solved within operational or planning time budgets.

Linear programming is a natural starting point for this investigation. LPs appear directly in dispatch models and indirectly inside larger workflows: as relaxations of mixed-integer problems, as subproblems in decomposition methods, as scenario blocks in stochastic models, and as repeated solves inside long-term planning tools. In PSR's own software ecosystem, this includes models built in JuMP, economic dispatch formulations for system operators, and decomposition schemes in which many related LPs must be solved or checked repeatedly.

For more than two decades, progress in solving large LPs has depended as much on algorithms as on hardware. The dominant paradigm for high-quality large-scale LP solution has been the interior-point method (IPM), which today is used primarily on CPUs. More recently, first-order primal-dual hybrid gradient methods (PDLP) have emerged as a credible alternative for very large problems. Instead of solving expensive Newton systems, these methods rely mainly on sparse matrix-vector products, an operation that maps naturally onto the massively parallel architecture of modern GPUs.

This article discusses the motivation for GPU-powered LP solvers, the types of power-system optimization problems where they may be especially relevant, and a benchmark developed with NVIDIA cuOpt, NVIDIA's GPU optimization library. The benchmark is a concrete case study: it shows how GPU-based algorithms behave on a large, structured dispatch LP and what that suggests for broader energy-analytics workflows.

Why GPUs for large LPs?

Two algorithmic paradigms are especially relevant for large-scale LPs:

Interior-point methods (IPMs): These remain the dominant approach for many large-scale LPs on CPUs. Each iteration requires the solution of a sparse Newton or KKT system, making sparse linear algebra the main computational bottleneck. GPU-accelerated IPM implementations are beginning to emerge, but sparse factorizations for the irregular matrices common in power-system optimization remain challenging to parallelize efficiently.

First-order primal-dual methods (PDLP): These methods form a more recent class of solvers in which each iteration is dominated by sparse matrix-vector products. They avoid factorization, and their core operations parallelize naturally on GPUs. The trade-off is that convergence can be sensitive to problem conditioning and to the requested tolerance.

The practical question is whether, for industrial-scale LPs, a first-order GPU solver can deliver a meaningful speed-up over a high-quality CPU IPM while still producing solutions of the quality required for analytical and operational use. The answer depends on problem size, numerical conditioning, tolerance requirements, and the surrounding workflow. GPUs are most attractive when the model is large enough to keep the device busy and when most of the computational work can be expressed as repeated sparse linear algebra.

The benchmark instance

To make the discussion concrete, we consider a day-ahead economic dispatch model of the Brazilian Interconnected System (SIN), operated by ONS. The model is formulated as a large-scale linear program, with no unit commitment binary variables. It represents thermal, renewable, hydro, and storage resources, with their operational characteristics captured through parameters such as reservoir volumes, inflows, ramping limits, and conversion efficiencies. The transmission network is modeled using an angle-based DC power flow formulation, while HVDC links are represented as controllable power injections.

In this formulation, bus voltage angles are explicit decision variables, and line flows are linked to angle differences through the linearized DC network equations. This representation is sparse and direct, making it a useful benchmark for evaluating how GPU-based first-order methods handle the large but structured LPs that arise in dispatch applications.

Methodology

The benchmark compares a CPU interior-point solver with the GPU PDLP solver in NVIDIA cuOpt. The two solvers belong to different algorithmic families and were configured at their respective default tolerances.

The CPU baseline is the HiGHS interior-point solver, a widely used open-source IPM. The GPU solver is NVIDIA cuOpt, configured at its default convergence tolerances for relative primal infeasibility, relative dual infeasibility, and relative duality gap. Experiments were performed on NVIDIA Grace Blackwell architecture. In the CPU runs, only the host CPU is used. In the GPU runs, the LP is solved on the GPU after presolve. The reported metric is solver-side solve time: the IPM phase for the CPU baseline and the PDLP phase for cuOpt.

To probe the scaling regime, the dispatch LP was run at five horizon lengths - 1, 2, 4, 12, and 24 hours - using the same network and resource data. The 24-hour case is the operationally relevant size, while the shorter horizons help identify the point at which the GPU advantage begins to dominate. The 24-hour instance reaches 757k rows, 1.42M columns, and 2.75M non-zeros.

Key results

The benchmark produces a clear scaling pattern, summarized in Figure 1.

First, on the operationally relevant 24-hour instance, the GPU PDLP solver is roughly three times faster than the HiGHS IPM baseline. cuOpt solves the 757k-row LP in 464 seconds, compared with 1,400 seconds for HiGHS, a 3.0x speed-up. At the 12-hour horizon, the speed-up is 1.8x (172 s vs. 312 s). The two solvers agree on the objective within roughly a 10^{-3} relative gap on every instance solved by both methods, which is consistent with the default first-order tolerance. The comparison therefore points

to a genuine computational advantage in the large-instance regime, while also making clear that feasibility and residual checks remain important when translating first-order solutions into operational use.

Second, the GPU advantage appears as the problem grows. Below the 12-hour horizon, the LP is still small enough for the CPU IPM to remain competitive or faster: at 1 hour horizon, HiGHS takes 3.3 s versus 14.2 s for cuOpt, and at 2 and 4 hours the two solvers are within a relatively small factor. This behavior is consistent with the underlying algorithms. PDLP iterations are individually inexpensive and composed almost entirely of sparse matrix-vector products, but the problem must be large enough to amortize fixed GPU launch and data-movement costs before the parallelism pays off.

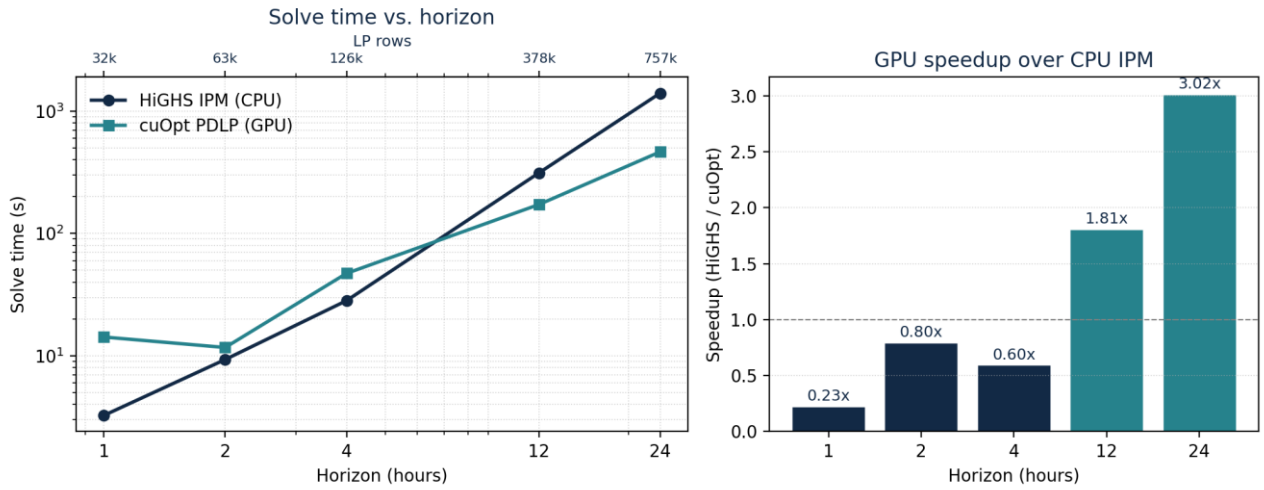


Figure 1 - Solve time (left, log--log) and GPU speed-up (right) vs. horizon length for the Brazilian dispatch LP. The CPU IPM (HiGHS) wins on the smallest instances; the GPU PDLP solver (cuOpt) takes over above the 4-hour horizon and reaches a 3 times advantage on the 24-hour, 757k-row instance.

What a faster dispatch LP enables

A speed-up on a single dispatch LP is useful in isolation, but the operational implications are broader than the headline number suggests. Dispatch models are solved many times per day and are often embedded inside iterative planning, security assessment, and uncertainty-analysis workflows.

Three concrete uses of the faster solve stand out:

More frequent re-dispatch: In intraday operation, the LP is re-solved as forecasts and system conditions change. A faster solve time can shorten the re-dispatch cycle, improving responsiveness to forecast errors and unplanned events.

More scenarios in stochastic and robust workflows: Stochastic dispatch and uncertainty-aware planning multiply the deterministic LP across many scenarios. A faster per-scenario solve can translate almost directly into more scenarios within the same wall-clock budget.

Higher model fidelity: Faster LPs make it more practical to include additional contingencies, finer time discretization, and more detailed network representations without exceeding the operational time budget.

The broader implication is that GPU acceleration is most valuable when the dispatch problem remains in a large, structured LP. Many operational and planning extensions preserve that structure: additional scenarios, finer temporal resolution, more detailed storage representation, and richer network detail can all increase problem size without necessarily changing the mathematical class of the model. In that

setting, the relevant question is not only whether one LP solves faster, but whether the solver makes a larger analytical workflow practical within the same time budget.

Discussion and conclusions

The benchmark answers a practical question within a broader research direction: can a GPU first-order LP solver deliver a meaningful speed-up on a production-shaped power-system LP? For this class of problem, the answer is yes. On the Brazilian day-ahead dispatch LP with an angle-based DC network formulation, cuOpt PDLP on current-generation NVIDIA hardware solves the 24-hour instance roughly three times faster than the HiGHS CPU interior-point method (464 seconds vs. 1400 seconds), with both solvers reporting an optimal solution and matching objective values within the default first-order tolerance.

Two qualifications are important. First, GPU acceleration does not dominate across all problem sizes. On small LPs, host overhead can outweigh the benefits of parallelism, and the CPU IPM remains the better choice. The GPU advantage materializes once the problem is large enough to keep the device effectively utilized. Second, PDLP is a first-order method and is sensitive to conditioning and tolerance requirements. Reaching tolerances significantly tighter than the default may be a different computational regime, where interior-point methods retain a structural advantage. This means the value of GPU-based PDLP should be assessed in relation to the accuracy level required by the workflow, not only by raw solve time.

The main conclusion is that GPU-based algorithms can change the computational envelope for large-scale optimization in energy analytics. They do not replace CPU interior-point methods in every regime, but they provide a strong alternative when the model is large, sparse, and dominated by operations that GPUs can execute efficiently. This matters because many of the most relevant modeling improvements in the sector increase LP size rather than changing the fundamental structure of the problem. Faster solutions of this core layer can therefore support more detailed, more frequently repeated, and more scenario-rich studies while keeping computation within practical limits.

For PSR and NVIDIA, the significance of the result is not only the specific speed-up reported in one dispatch benchmark. It is evidence that GPU-powered LP algorithms are becoming relevant to the real optimization workloads behind power-system planning and operation. As models continue to grow in temporal resolution, spatial detail, and uncertainty representation, the ability to exploit GPU architecture will become an increasingly important part of the optimization toolbox.