Monte Carlo Simulation Support in LunaSim: A Browser-Based System Dynamics Modeling Platform for Sensitivity Analysis

Akash Saran*†, Arjun Mujumdar*, Ryan Chung*, Caleb Wen*, Ishan Khetarpal, and Mark R. Estep
Poolesville High School, Poolesville, Maryland, USA
*Equal Contribution. †Corresponding Author: akashsaran.ssp@gmail.com

***Abstract*—LunaSim is a lightweight, fully browser-based, open-source system dynamics modeling platform extended here with Monte Carlo simulation support, three new stock types, and live real-world data integration through external APIs. Parallel ensemble execution is achieved via the Web Worker API, enabling execution time reductions of up to 68.5%. Three new stock types (Microwave, Conveyor, and Queue) each expose a named structural parameter for direct uncertainty sampling without modifying model structure. Furthermore, live weather, macroeconomic, and financial market data can be injected directly into model equations via bracketed API references, allowing confidence bands grounded in current real-world conditions. Simulated percentile bands were compared against analytically derived outcomes, and qualitative behavior was validated across all three new stock types, establishing LunaSim as an accessible tool for probabilistic system dynamics modeling.**

***Index Terms*—Monte Carlo simulation, Uncertainty Analysis, Web-Based Modeling, Real-World Data**

I. Introduction

System Dynamics (SD) modeling is the process of breaking down a complex system into simpler elements and simulating their interactions to predict how the system evolves. Graphical representations such as stock-and-flow diagrams and causal loop diagrams are tools that aid in understanding the complex relationships among system components, but SD can equally be expressed through equations that capture feedback, delays, and accumulations without any graphical annotation. The fundamental building blocks of SD are stocks, which represent quantities that accumulate over time, and flows, which regulate those accumulations. Variables and influences exist to group intermediate calculations and indicate directional relationships between elements. This stock-and-flow architecture is consistent across the entire SD software ecosystem; tools such as STELLA [1], Vensim [2], and Berkeley Madonna [3] all share these same primitive element types.

LunaSim [4] is a lightweight, browser-based SD modeling platform developed entirely by high school students at Poolesville High School. It was originally created after a school-wide technology upgrade rendered the commercial modeling software used in the school's introductory computer science curriculum inoperable on Windows computers and school-issued Chromebooks, which motivated students to build a free, open-source alternative that runs directly in any standard web browser. Fig. 1 shows the LunaSim model editor, where stock-and-flow diagrams are constructed, and Figs. 2 and 3 show

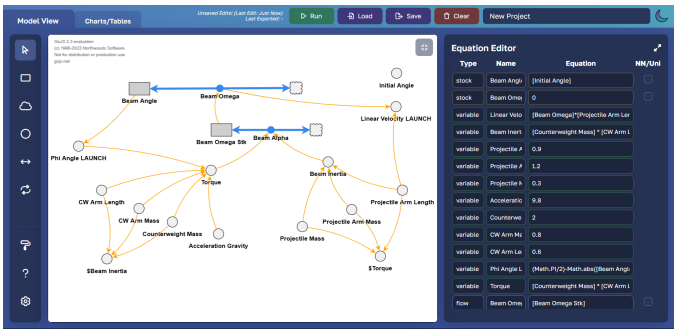

Fig. 1: The LunaSim model editor, showing an example counterweight-trebuchet model built from stocks and variables such as beam angle, beam angular velocity, counterweight mass, projectile mass, and torque, along with their equations in the Equation Editor panel.

Fig. 2: Example LunaSim output showing the position and velocity of an oscillating mass-spring system over a 10-second simulation.

example simulation results visualized as graphs within the platform. The work presented in this paper extends LunaSim with the Monte Carlo, new stock type, and live API integration capabilities described below.

A natural limitation of deterministic SD models is their sensitivity to parameter uncertainty. In models that represent real-world scenarios, inputs such as growth rates, decay constants, and initial conditions are estimated from noisy empirical data. When a model is run with fixed estimates for these parameters, the resulting simulation represents only one possible rendition of the system, which can be misleading. Monte Carlo simulation [5] addresses this by executing a large ensemble of runs, each with parameters sampled from user-specified probability distributions, aggregating the results

Fig. 3: Example LunaSim output showing the growth of a modeled algae population over a 10-second simulation.

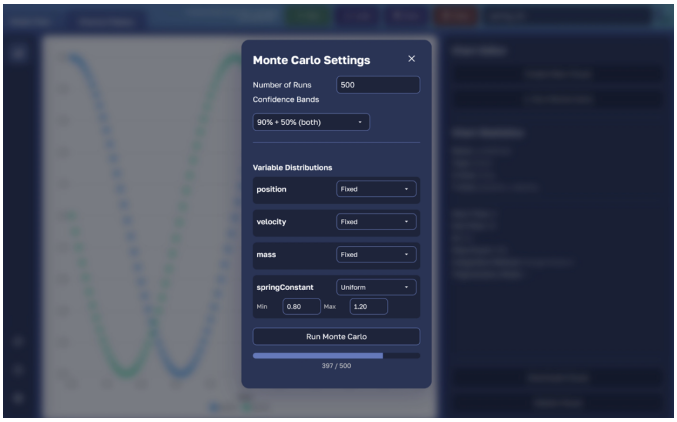

Fig. 4: Monte Carlo Settings popup showing configuration options including number of runs, confidence band selection, and variable distribution assignments.

into probabilistic confidence bands that characterize the full range of plausible system behavior [6]. In fields such as public health epidemiology [7], ecological modeling [8], and systems engineering [9], understanding the envelope of possible outcomes is equally as important as estimating the true outcome itself. Effective Monte Carlo simulation also requires that model parameters be structured in a way that makes them directly accessible for sampling; when uncertain parameters are embedded within user-written equations, isolating them becomes cumbersome. Dedicated stock types with named structural parameters simplify this process directly. Additionally, grounding confidence bands in live real-world data rather than assumed values makes uncertainty analysis more relevant to current conditions. While existing browser-based SD platforms lack these capabilities, LunaSim is extended with Monte Carlo simulation support, three new stock types, and live API integration to address this gap.

II. Related Work

Vensim PLE+ [10] is the most widely used system dynamics tool with built-in Monte Carlo sensitivity analysis. Users define probability distributions for selected parameters through a sensitivity simulation setup, and Vensim PLE+ executes an ensemble of runs, displaying the resulting output as percentile-based confidence bands. However, the need for a desktop installation and a commercial license presents obstacles to its use in educational and collaborative environments. LunaSim [4], on the other hand, is a lightweight, completely open-source SD modeling platform that requires no installation and operates entirely within a web browser, making it especially well-suited for classroom and collaborative use.

III. Methods and Results

A. Monte Carlo Simulations

Monte Carlo support is implemented so that individual simulation runs can be dispatched to parallel worker threads using the Web Worker API [11], substantially improving performance for large ensemble sizes.

The Monte Carlo workflow begins when the user opens the Monte Carlo popup (Fig. 4) and specifies the number of runs N, selects a confidence band setting (90%, 95%, 99%, or a combined 90%+50% display), and assigns a probability distribution to one or more model variables. Four distribution types are supported: normal, sampled using the Box–Muller transform [12]; uniform; triangular [13]; and fixed. These four were selected to cover the most common uncertainty characterizations encountered in practice: normal for quantities with symmetric measurement error, uniform for parameters with known bounds but no preferred value, triangular for cases where a most-likely estimate exists alongside minimum and maximum bounds, and fixed for parameters that should remain deterministic while others are varied. The user's selected distribution assignments are then associated with their corresponding variables. When the user initiates a run, N simulation runs are dispatched in parallel, each receiving the base model and a parameter vector sampled from the user's specified distributions. When all runs have completed, results are aggregated and percentiles are computed at each timestep. The following percentile levels are computed: p0.5, p2.5, p5, p25, p50, p75, p95, p97.5, and p99.5, along with the mean and standard deviation across all runs. Monte Carlo results are displayed in a new chart tab (Fig. 5), where the selected confidence band is rendered as a filled range-area series with the p50 median overlaid as a line series. Users can switch between tracked variables without leaving the tab. Monte Carlo results are not persisted across page refreshes and must be regenerated if the page is reloaded.

B. Integration with New Stocks

LunaSim introduces three new stock types (Microwaves, Conveyors, and Queues), each of which exposes a named structural parameter that the Monte Carlo engine can sample directly. The primary value of these new stock types over the primitive stock is found in the Monte Carlo simulation. In a primitive stock driven by a user-written equation, uncertain parameters and timing logic are entangled within a single expression. For example, a modeler wishing to vary the duration of a fixed-value phase must manually locate and extract that value from a conditional formula before the Monte Carlo engine can treat it as a free parameter. This coupling between structure

Fig. 5: Results chart tab displaying 90%+50% confidence bands over 500 runs of a spring model with `springConstant` drawn from a uniform distribution.

and parameter makes targeted uncertainty analysis cumbersome. By introducing these stocks, which encapsulate fixed-duration behavior (Microwave), pipeline transit (Conveyor), and capacity overflow (Queue) with respective parameters `CookTime`, `TransitTime`, and `CapacityLimit`, the Monte Carlo engine can vary these parameters directly and independently without modifying the surrounding model structure.

1) Microwaves: A "Microwave" is a stock that takes a constant value for a specified amount of time from the start of the simulation until it becomes zero. The user can input a value for the `CookTime`, which will specify the time at which the respective Microwave's value will become zero. Microwaves in LunaSim are analogous to ovens in STELLA, which follow the same behavior: maintaining a fixed value for a prescribed duration before resetting to zero [14].

While LunaSim allows a similar effect to be replicated using a conditional expression on the built-in `[TIME]` variable, the Microwave component still offers advantages. Most importantly, naming it a "Microwave" with a "CookTime" parameter makes the intent of the component self-evident with an analogy: just as a microwave cooks something for a fixed time and then stops, the component does the same thing to a value in the simulation. A modeler reading the diagram doesn't need to parse a formula to understand that this stock holds a value for a set duration and then expires.

In the context of Monte Carlo simulation, the `CookTime` parameter is the natural target for uncertainty sampling. Consider a drug efficacy model in which a treatment is sustained for a fixed therapeutic window before wearing off. The exact duration of this window is rarely known with certainty and is typically estimated from clinical trial data with associated variance. Using a Microwave stock to represent the treatment effect, and assuming a normal distribution for its `CookTime`, allows the Monte Carlo engine to explore possible therapeutic durations. The confidence bands reveal how sensitive the downstream health outcomes are to the uncertainty in the treatment window, without altering any other part of the model.

To demonstrate this, we construct a minimal drug efficacy model consisting of a single Microwave stock with no inflows or outflows; the full parameter set is listed in Table I. The stock retains its starting value throughout the `CookTime`, subsequently dropping to zero, eliminating the need for extra components.

TABLE I: Microwave Model Parameters (`TreatmentEffect`)

Parameter	Value
Stock Type	Microwave
Stock Name	TreatmentEffect
Initial Value	100
CookTime	10
Start Time	0
End Time	15
dt	0.1
MC Runs	200
Sampled Parameter	CookTime
Distribution	Normal
Mean	10
Std Dev	1.5
Confidence Band	90%

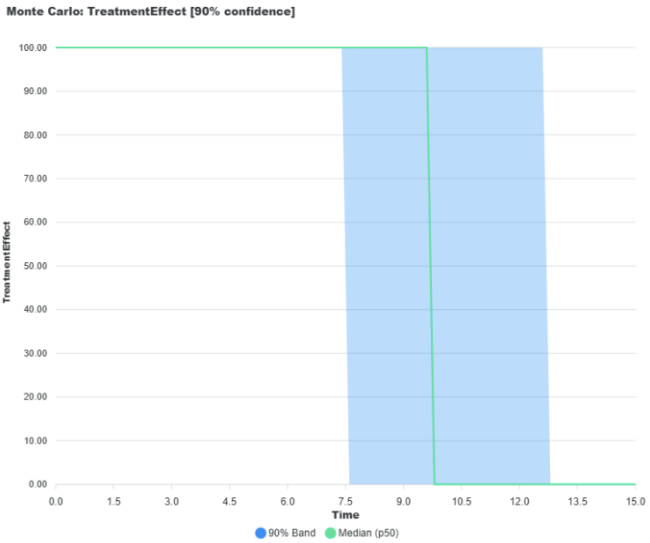

Fig. 6: Monte Carlo simulation of `TreatmentEffect` under normally distributed `CookTime` uncertainty, displaying a 90% confidence band over 200 runs.

Figure 6 shows the resulting Monte Carlo output. From t=0 to approximately t=7.5, the green median line and the light blue 90% confidence band are entirely coincident at 100, indicating that across all 200 runs the treatment effect remains at full strength with no variation. Around t=7.5, the lower edge of the blue band begins to drop. This indicates runs where a shorter-than-usual `CookTime` caused the treatment to end early. The upper edge, however, doesn't reach zero until about t=12.5. This represents runs where a longer `CookTime` allowed the effect to continue beyond the average. The width of the transition area, spanning roughly from t=7.5 to t=12.5 (about 5 time units), directly shows how a standard deviation of 1.5 in `CookTime` leads to uncertainty about when the therapeutic effect stops.

Fig. 7: LunaSim model editor showing the `InTransit` Conveyor stock connected to `ReceivedInventory` via `flow1`.

2) Conveyors: A "Conveyor" is a stock whose value decreases over time until reaching zero, in a specified manner. The user can choose for the respective Conveyor's value to decrease linearly or exponentially, reaching zero at the `TransitTime` specified by the user. Conveyors in LunaSim are analogous to conveyors in STELLA [14], which model the same concept of material or value moving through a pipeline and being released after a fixed transit duration. Just like in STELLA, a conveyor moves quantities through in a first-in, first-out fashion, emptying at a rate determined by its length and transit time. Just as a factory conveyor belt carries items from one end to the other over a fixed amount of time, gradually emptying as items fall off the end, the Conveyor stock carries a value through the simulation and gradually depletes it until nothing remains at the `TransitTime`.

In the context of Monte Carlo simulation, the `TransitTime` parameter directly represents pipeline delay: a quantity that is frequently uncertain in real systems. For example, in a supply chain model, raw materials traveling from a supplier to a manufacturing site experience transit times that are not constant. These durations vary due to factors like routing decisions, customs clearance, and the consistency of transportation providers, all of which introduce variability into the system. By representing the in-transit inventory as a Conveyor stock and assigning a triangular distribution to its `TransitTime`, the Monte Carlo engine can simulate the range of possible delivery schedules. The resulting confidence bands on downstream inventory and production stocks reveal how sensitive the supply chain is to transit time variability, informing decisions about safety stock levels and reorder timing.

To demonstrate this, we construct a minimal supply chain model consisting of a Conveyor stock connected to a downstream regular stock via a flow; the full parameter set is listed in Table II, and the resulting model is shown in Fig. 7. A cloud is placed at the left end of the Conveyor representing an external supplier as the material source. The Conveyor's transit-completion outflow connects directly to `ReceivedInventory`, and no valve equation is needed on the transit outflow as the engine computes the delivery rate automatically from the pipeline.

Figure 8 shows the resulting Monte Carlo output. At t=0, all runs begin at exactly 500 with zero spread. The light blue 90% band and the teal 50% band fan out progressively as time advances: by t=7.5, the 90% band spans roughly 280 to 460

TABLE II: Conveyor Model Parameters (`InTransit`)

Parameter	Value
Stock Type	Conveyor
Stock Name	InTransit
Initial Value	500
TransitTime	20
Downstream Stock	ReceivedInventory
Downstream Stock Initial Value	0
Start Time	0
End Time	15
dt	0.1
MC Runs	200
Sampled Parameter	TransitTime
Distribution	Triangular
Min	12
Mode	20
Max	35
Confidence Band	90% + 50%

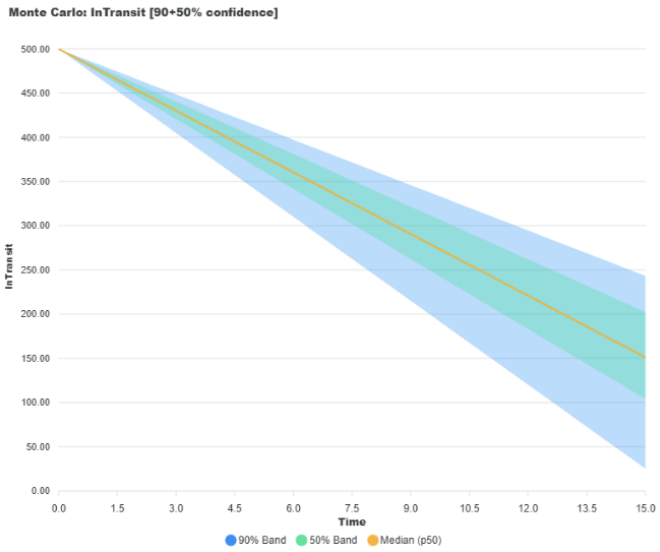

Fig. 8: Monte Carlo simulation of `InTransit` under triangularly distributed `TransitTime` uncertainty, displaying a combined 90%+50% confidence band over 200 runs.

while the tighter 50% inner band spans roughly 360 to 430, reflecting the triangular distribution's concentration around its mode of 20. The asymmetry of the blue band, wider below the median than above, reflects the right skew of the triangular distribution, as the maximum of 35 is further from the mode than the minimum of 12, meaning slower-draining runs produce a longer tail of higher inventory values. The orange median tracks a clean linear drain through the center of both bands, reaching approximately 150 at t=15.

3) Queues: A "Queue" is a stock that accumulates, similar to an ordinary stock; however, once it reaches its user-specified `CapacityLimit`, its value will remain capped, meaning that any additional inflow will be redirected to a separate outflow indicated by the red arrow. The key difference between LunaSim and STELLA's implementation is that STELLA's queue is ordered; it tracks individual entities moving through

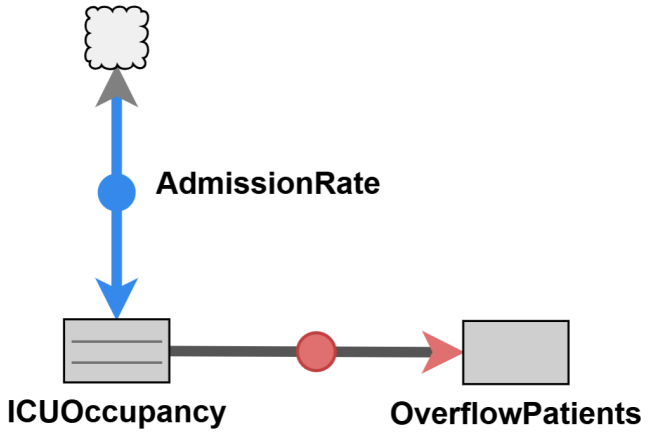

Fig. 9: LunaSim model editor showing the `ICUOccupancy` Queue stock with `AdmissionRate` inflow and `OverflowPatients` overflow outflow.

in a first-in, first-out fashion, preserving the sequence of arrivals. LunaSim's Queue, by contrast, treats the stock as an aggregate value rather than a sequence of discrete entities, making it simpler and more in line with the continuous system dynamics paradigm while still faithfully enforcing the capacity and overflow behavior.

In the context of Monte Carlo simulation, the `CapacityLimit` parameter is a natural target of structural uncertainty. Consider a hospital resource model in which an ICU ward has a finite bed capacity. In practice, effective capacity is not a fixed number; it fluctuates due to staffing constraints, equipment availability, and administrative factors. By modeling ICU occupancy as a queue and assigning a uniform distribution to its `CapacityLimit` within a reasonable range, the Monte Carlo engine can simulate the variability of overflow, which is the number of patients redirected to other care options (shown by the red arrow outflow), across different capacity scenarios. The resulting confidence intervals for overflow volume directly measure the risk of ICU saturation, providing hospital planners with relevant uncertainty estimates for decision-making, without requiring any changes to the model's structure.

To demonstrate this, we construct a minimal ICU resource model consisting of a Queue stock with one regular inflow and one overflow outflow; the full parameter set is listed in Table III, and the resulting model is shown in Fig. 9. A cloud is placed on the left as the source of incoming patients. The inflow valve `AdmissionRate` carries a constant equation of 5, representing a steady admission rate. An overflow arrow connects `ICUOccupancy` to `OverflowPatients` (this is drawn using the overflow tool, not a regular flow). No valve equation is needed on the overflow arrow; the engine redirects excess automatically when the queue hits its `CapacityLimit`.

Figure 10 shows the resulting Monte Carlo output. From t=0 to approximately t=7, the green median line and the light blue 95% confidence band are fully coincident, rising linearly

TABLE III: Queue Model Parameters (`ICUOccupancy`)

Parameter	Value
Stock Type	Queue
Stock Name	ICUOccupancy
Initial Value	0
Capacity	50
Inflow Valve Name	AdmissionRate
Inflow Valve Equation	5
Inflow Source	Cloud
Overflow Destination Stock	OverflowPatients
OverflowPatients Initial Value	0
Start Time	0
End Time	10
dt	0.1
MC Runs	200
Sampled Parameter	CapacityLimit
Distribution	Uniform
Min	35
Max	65
Confidence Band	95%

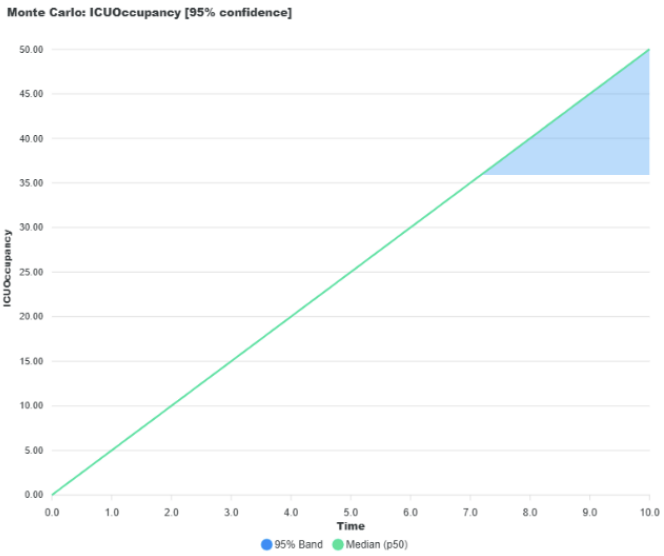

Fig. 10: Monte Carlo simulation of `ICUOccupancy` under uniformly distributed `CapacityLimit` uncertainty, displaying a 95% confidence band over 200 runs.

from 0 at a rate of 5 units per time step, as no run has yet reached its capacity ceiling. At approximately t=7, the lower bound of the band begins to diverge downward, reflecting runs in which a lower sampled capacity has caused the queue to cap and stop accumulating, while the upper bound continues rising to 50 at t=10, representing runs in which the sampled capacity was not reached within the simulation window. The shape of the band (a clean linear ramp that abruptly fans open only at saturation) is the characteristic signature of capacity uncertainty in a queue: deterministic behavior below the threshold and divergence only at the point of constraint.

C. Integration with Live APIs

LunaSim incorporates three external data sources (weather, macroeconomic, and financial market data) chosen to represent

categories of real-world parameters commonly used as inputs in system dynamics models. Weather data, sourced from Open-Meteo [15], underpins environmental and epidemiological models in which factors such as temperature, humidity, and precipitation directly affect system behavior. Macroeconomic indicators, obtained from the World Bank Indicators API [16], inform economic and policy models; these indicators, including GDP, inflation, and unemployment rates, define the structural environment of the system. Financial market data, provided by Finnhub [17], facilitates portfolio and investment models in which live equity prices anchor return rate calculations to prevailing market conditions. Location information for weather queries is resolved through Nominatim [18]. This setup allows modelers to tie simulation parameters directly to real-world figures rather than fixed assumed values, producing models that reflect current conditions at runtime.

Each API is accessed using a specific syntax within any equation field. Weather variables are referenced as `[variable][zip code]`, where `variable` can be `wind`, `temp`, `humidity`, `aq`, or `precip`, and the zip code is any valid US zip code. The zip code is first converted into geographic coordinates through Nominatim's geocoding service, which are then used by Open-Meteo to obtain current meteorological data. Macroeconomic indicators are denoted in the format `[variable][country]`, where `variable` represents one of the following: `gdp`, `inflation`, `unemployment`, `population`, `gnipc`, or `trade balance`, and `country` is identified by an ISO 3166-1 alpha-2 country code [19]. Financial data is referenced as `[stock][ticker]`, where `ticker` is any valid equity symbol supported by Finnhub, returning the current market price of that security.

Monte Carlo simulation can be used in conjunction with these API integrations to combine live real-world data with parameter-level uncertainty analysis. Because API values are resolved once at the start of each simulation run and held constant across all timesteps, the Monte Carlo engine treats each resolved value as a fixed seed for that run. A modeler can then assign a distribution to any parameter that depends on or relates to that seeded value. For example, a portfolio model (Fig. 11) using a live stock price from `[stock][AAPL]` can assign a normal distribution to its growth rate, allowing for the study of how market volatility changes over time. Similarly, a disease model using live humidity data from `[humidity][10001]` can sample its recovery rate from a triangular distribution, reflecting clinical uncertainty. As a result, LunaSim models can produce confidence bands grounded in current real-world conditions rather than just hypothetical parameter ranges. The full parameter set for this portfolio model is listed in Table IV, and Fig. 12 shows the resulting Monte Carlo output, combining uncertainty in both the annual return rate and the withdrawal rate.

IV. Testing & Validation

To validate numerical correctness, the model was simplified to a single sampled parameter: `annualReturnRate`

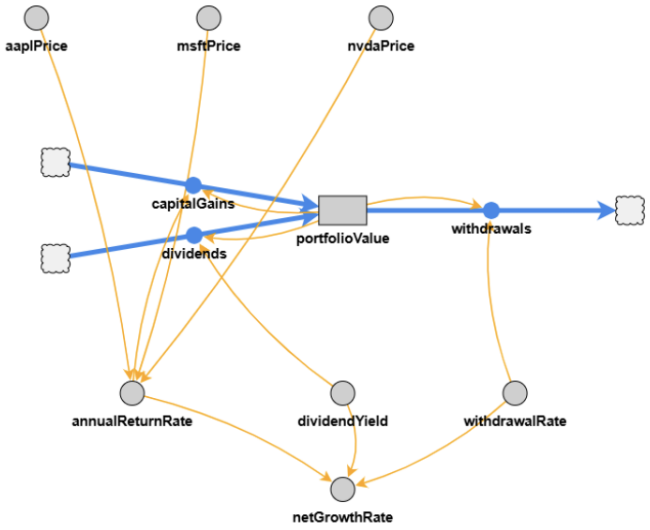

Fig. 11: LunaSim diagram of Long-Term Investment Growth Model, incorporating live equity prices for AAPL, MSFT, and NVDA via the Finnhub stock market API to dynamically adjust annual return rate.

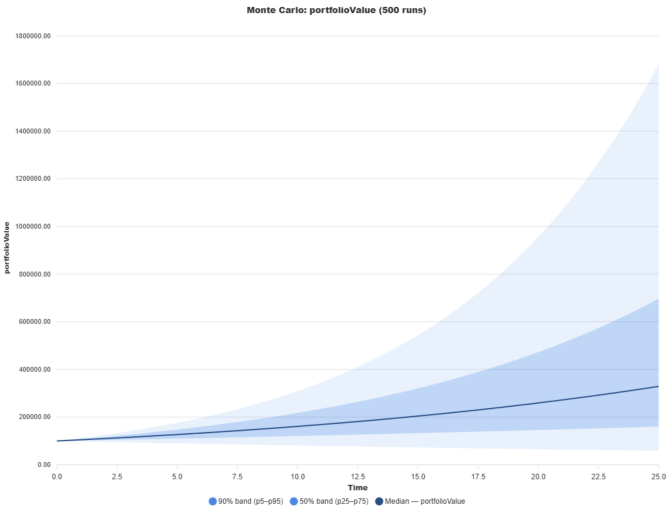

Fig. 12: Monte Carlo simulation of `portfolioValue` under normally distributed `annualReturnRate` uncertainty (mean 0.07, std dev 0.04) and uniform `withdrawalRate` uncertainty (min 0.03, max 0.06), displaying a 90%+50% confidence band over 500 runs.

drawn from $\mathcal{N}(0.07, 0.04)$, with `withdrawalRate` and `dividendYield` held fixed at 0.04 and 0.015 respectively, yielding the closed-form analytical solution:

$$P_p(t) = 100{,}000 \times e^{(0.045+0.04\times z_p)\times t} \quad (1)$$

where z_p is the standard normal quantile for percentile p. Table V compares simulated percentiles from a 500-run ensemble against analytical values at $t = 5, 10, 15, 20,$ and 25. The maximum absolute percentage deviation across all percentile levels and timesteps was 6.84%, observed at p50 and t=25. The deviation grows monotonically with time, consistent with the expected accumulation of ensemble sampling variance over a longer simulation horizon rather than a systematic

TABLE IV: API-Integrated Portfolio Model Parameters (`portfolioValue`)

Parameter	Value
Stock Name	portfolioValue
Initial Value	100,000
annualReturnRate	0.07 + market adjustment (AAPL, MSFT, NVDA)
dividendYield	0.015
withdrawalRate	0.04
aaplPrice	`[stock][AAPL]` (live)
msftPrice	`[stock][MSFT]` (live)
nvdaPrice	`[stock][NVDA]` (live)
Start Time	0
End Time	25
dt	0.1
MC Runs	500
Sampled Parameter 1	annualReturnRate
Distribution 1	Normal
Mean	0.07
Std Dev	0.04
Sampled Parameter 2	withdrawalRate
Distribution 2	Uniform
Min	0.03
Max	0.06
Confidence Band	90% + 50%

TABLE V: Analytical vs. Simulated Percentiles: Investment Portfolio Model (500 runs)

t	p5 An.	p5 Sim.	p50 An.	p50 Sim.	p95 An.	p95 Sim.
5	90,123	89,931	125,232	126,900	174,030	175,814
10	81,221	80,876	156,831	161,035	302,829	309,107
15	73,198	72,733	196,408	204,353	526,996	543,455
20	65,965	65,410	245,960	259,324	917,063	955,474
25	59,453	58,824	308,028	329,082	1,595,800	1,679,868

numerical error in the integration engine. The p5 column exhibited the smallest deviations throughout, remaining below 1.1% across all timesteps.

Beyond numerical correctness, parallel execution performance was evaluated across three ensemble sizes. The number of Web Workers spawned scales with the number of logical cores available on the host machine, up to a maximum of 10; a machine with fewer than 10 cores spawns one worker per core, while any machine with 10 or more cores spawns exactly 10 workers. Each configuration was tested across six trials total: three sequential and three parallel, and the reported values are the averages of those three trials for each method. All benchmarks were conducted on a Lenovo ThinkPad X9-15 Gen 1 (model 21Q6002BUS) equipped with an Intel Core Ultra 7 268V (8 cores, 8 logical processors, Lunar Lake architecture), 32 GB LPDDR5x RAM, running Microsoft Edge on Windows 11 Pro; this machine spawned 8 workers for all benchmark runs. The results are summarized in Table VI.

V. Discussion

The analytical comparison in Table V confirms that simulated percentiles tracked their closed-form counterparts closely, with a maximum deviation of 6.84% at p50 and t=25. This deviation grew monotonically with time rather than accumulating randomly, indicating that the error reflects ensemble sampling

TABLE VI: Monte Carlo Execution Time: Sequential vs. Web Worker (Investment Portfolio Model, 250 Timesteps, 8 Workers)

Runs (N)	Seq. (s)	Web Workers Used	Worker (s)	Time Reduction (%)
100	4.76	8	2.00	57.9
500	24.38	8	7.85	67.8
1,000	50.48	8	15.91	68.5

variance over a longer horizon rather than a systematic flaw in the integration engine, and the p5 column in particular remained below 1.1% across all timesteps. Web Worker parallelization reduced execution time by up to 68.5% at 1,000 runs; the benefit of parallelization was proportionally smaller at lower run counts, where worker initialization overhead represents a larger fraction of total execution time.

The new stock types proved particularly useful for Monte Carlo analysis because `CookTime`, `TransitTime`, and `CapacityLimit` are exposed as standalone parameters that the Monte Carlo engine can vary directly without requiring the modeler to rewrite any equations. The confidence bands produced across all three stock type demonstrations had shapes consistent with their input distributions: the Microwave showed a clean expiry window, the Conveyor showed progressive fanning driven by the triangular distribution's right skew, and the Queue showed deterministic behavior up until the point of saturation. The API integrations allow confidence bands to be grounded in current real-world conditions at runtime, enabling uncertainty analysis that reflects the actual state of the system being modeled rather than assumed parameter values.

Several limitations remain: results are not persisted across page refreshes; parameters cannot be sampled with correlations between them, meaning that in systems where quantities such as transit time and warehouse capacity are positively correlated, independent sampling may understate joint uncertainty; and external API calls can fail or be rate-limited, which may affect reproducibility.

VI. Conclusion

Monte Carlo simulation support, three new stock types, and live API integration were added to LunaSim, bringing probabilistic sensitivity analysis grounded in real-world data directly into the browser. Simulated percentile bands were validated against closed-form analytical solutions, and the observed deviations, consistently below 6.84% across all timesteps and percentile levels, are attributable to ensemble sampling variance rather than any systematic integration error. The Web Worker implementation reduced execution time by up to 68.5%, the new stock types made uncertainty sampling more direct and interpretable by exposing named structural parameters, and the weather, macroeconomic, and financial API integrations allow model parameters to be seeded from live data sources at runtime. Together these features bring capabilities that previously required commercial desktop software into a free, browser-based environment built for educational and collaborative use. Future work could address correlated parameter sampling to better capture joint uncertainty

in multi-parameter models, persistent storage of Monte Carlo results across page sessions, and expanded API coverage to support a broader range of modeling domains.

VII. Link to LunaSim

LunaSim's source code is publicly available at https://github.com/PHS-SMCS/LunaSim, and a live version of the application can be accessed at https://lunasim.org.

References

[1] STELLA Architect. Version 2.0.3, ISEE Systems. Available: https://www.iseesystems.com/store/products/stella-architect.aspx

[2] Vensim. Ventana Systems. Available: https://vensim.com/

[3] Berkeley Madonna. Berkeley Madonna. Available: https://berkeleymadonna.myshopify.com/

[4] K. S. Vedula, S. Simms, A. Patil, I. Khetarpal, and M. R. Estep, "LunaSim: A Lightweight, Web-Based, Open-Source System Dynamics Modeling Software," in *Proc. 2024 International System Dynamics Conference*, 2024. Available: https://proceedings.systemdynamics.org/2024/papers/P1049.pdf

[5] N. Metropolis and S. Ulam, "The Monte Carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[6] A. Saltelli et al., *Global Sensitivity Analysis: The Primer*. Wiley, 2008.

[7] R. Tlucherowska et al., "Method 'Monte Carlo' in Healthcare," PMC/MDPI, 2024. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11230067/

[8] T. J. Fullman et al., "Simulation modeling accounts for uncertainty while quantifying ecological effects of development alternatives," *Ecosphere*, vol. 12, no. 5, 2021. Available: https://esajournals.onlinelibrary.wiley.com/doi/10.1002/ecs2.3530

[9] E. Zio and N. Pedroni, "Monte Carlo Simulation-based Sensitivity Analysis of the model of a Thermal-Hydraulic Passive System," *Reliability Engineering and System Safety*, vol. 107, pp. 90–106, 2012.

[10] Ventana Systems, Vensim PLE+, 2023. Available: https://vensim.com/vensim-ple-plus/

[11] Mozilla Developer Network, "Web Workers API," 2024. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

[12] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.

[13] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, vol. 2, 2nd ed. Wiley, 1994.

[14] Isee Systems, "STELLA Architect Reference Manual," isee systems, 2024. Available: https://www.iseesystems.com/resources/help/

[15] Open-Meteo, "Open-Meteo Free Weather API," 2024. Available: https://open-meteo.com

[16] World Bank, "World Bank Open Data — Indicators API," 2024. Available: https://data.worldbank.org/indicator

[17] Finnhub, "Finnhub Stock Market API," 2024. Available: https://finnhub.io

[18] Nominatim, "Nominatim Geocoding API," OpenStreetMap Foundation, 2024. Available: https://nominatim.org

[19] International Organization for Standardization, "ISO 3166-1: Codes for the representation of names of countries and their subdivisions — Part 1: Country codes," ISO, 2020.