MONTE CARLO AIRCRAFT NOISE SIMULATION DASHBOARD

A probabilistic aeroacoustic simulation tool to quantify uncertainty in aircraft noise predictions.

MATLAB	App Designer	Monte Carlo	Parallel Computing	Statistics / FFT	Pratt & Whitney Canada

10^5+ Simulation runs	**8** Input parameters	**~10×** Runtime speedup	**80%** Confidence band

PROBLEM

Why this tool was needed

The aeroacoustics team at Pratt & Whitney Canada inherited uncertainty sigma values from RTX Hartford that had never been independently verified. The existing workflow ran one-off MATLAB scripts sequentially, one dataset at a time, with manual parameter perturbations standing in for real stochastic analysis. A full validation sweep took 4.5 to 5 hours, which made rapid iteration impractical.

There were two problems to solve. First, accuracy: stakeholders needed EPNL values they could trust against ICAO noise standards, and no one had confirmed whether the inherited sigmas were correct. Second, throughput: the team needed to run multiple datasets concurrently without waiting hours between results.

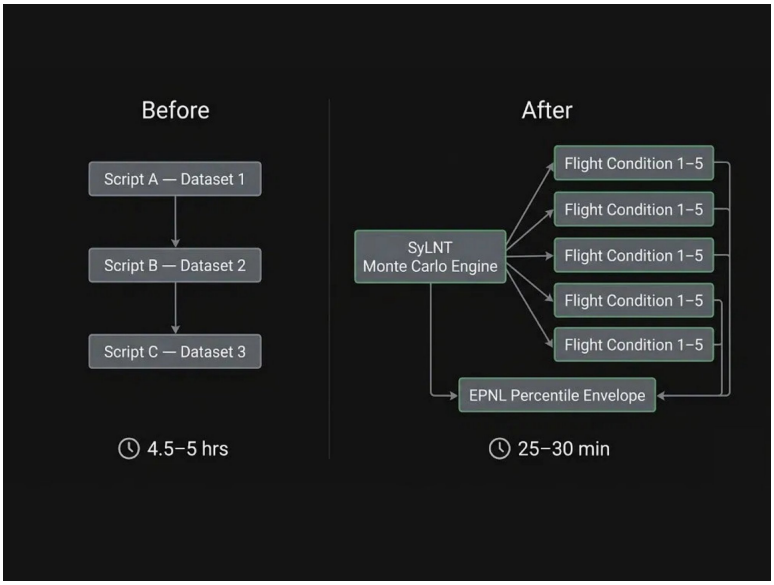


Fig. 1 — Prior sequential script workflow (left) vs. parallelized Monte Carlo architecture (right)

ARCHITECTURE

How the engine works

The tool wraps the team's proprietary deterministic noise model, which simulates individual engine components under specified flight conditions and outputs cumulative EPNL. The Monte Carlo engine calls that model repeatedly with independently sampled inputs each time, then aggregates all EPNL outputs into a statistical distribution.

01	02	03	04	05
Input distributions	Independent sampling	Noise model dispatch	EPNL aggregation	Percentile envelope

Each of the 8 input parameters draws an independent random sample per iteration. This decoupling was a critical fix. An earlier version used a single uniform draw applied to all variables at once, which created artificial cross-component correlation and narrowed the variance bounds in a way that did not reflect reality.

Convergence is verified by tracking the running variance on the 10th and 90th percentile bounds across successive iteration blocks. Once the delta between blocks drops below 0.05 dB, the run is considered statistically converged. The baseline is N >= 10,000 iterations per flight condition across 5 calibrated conditions, giving 50,000 total evaluations per sweep.

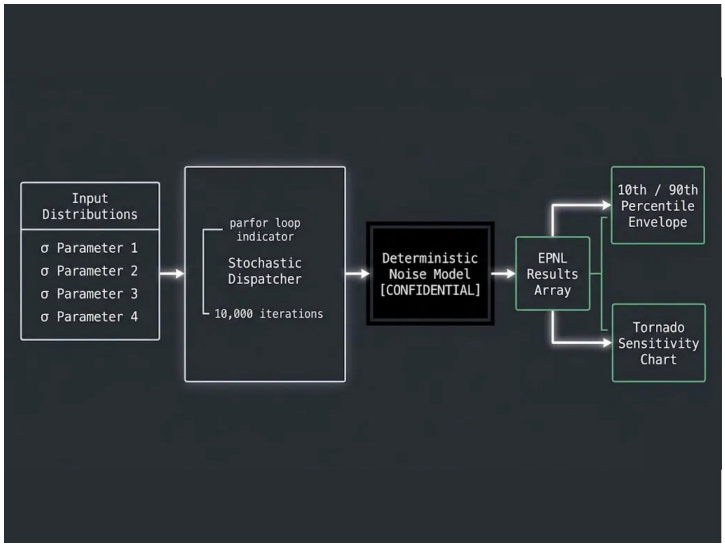


Fig. 2 — Engine architecture: input distributions → stochastic dispatcher → confidential noise model → EPNL outputs

KEY DECISIONS

Engineering tradeoffs

Non-parametric percentile envelope over Gaussian assumption

EPNL is computed from logarithmic combinations of frequency spectra, tone corrections, and duration allowances, which makes it inherently non-linear. Using a Gaussian assumption with mu +/- sigma risks miscalculating certification margins when the output distribution has skewness or heavy tails. The tool checks this dynamically each run using sample skewness, excess kurtosis, and a Jarque-Bera or Kolmogorov-Smirnov test. When non-Gaussianity is confirmed, raw non-parametric percentiles at the 10th and 90th are used, which is the only defensible method for extracting a valid 80% confidence envelope.

Independent per-variable sampling — fixing the coupling bug

The first version drew one pseudo-random value per iteration and applied it to all input components simultaneously. Output variances looked too narrow during data analysis, which is what flagged the problem. Switching to a unique independent draw per sub-component fixed the artificial correlation between physically unrelated variables. Variance bounds expanded appropriately in specific flight conditions and the unphysical worst-case peaks disappeared. In hindsight, a startup correlation matrix check would have caught this before it reached a production sweep.

Cursor-based measurement over a fixed average Delta readout

A single average Delta value is easy to implement but not very useful in practice. In multi-condition sweeps, the nominal regression line can diverge from flight test data at specific thrust settings while looking fine on average. A bulk metric hides that. The interface was redesigned around a draggable cursor built directly into the App Designer axes so engineers can probe any point on the EPNL distribution and read the exact Delta at that location. This came directly from feedback from the project lead and the aeroacoustics team manager.

Parallelizing the inner iteration loop, not the outer file loop

The 5 flight condition files are the outer loop. The 10,000 stochastic iterations are the inner loop, parallelized with MATLAB's parfor. Running parallel reads across the 5 files would have caused race conditions with the YS parser, which handles legacy text-based input. Instead, each worker received sliced input arrays in memory and workspace cleanup ran automatically after each iteration to avoid memory leaks over long batch sweeps.

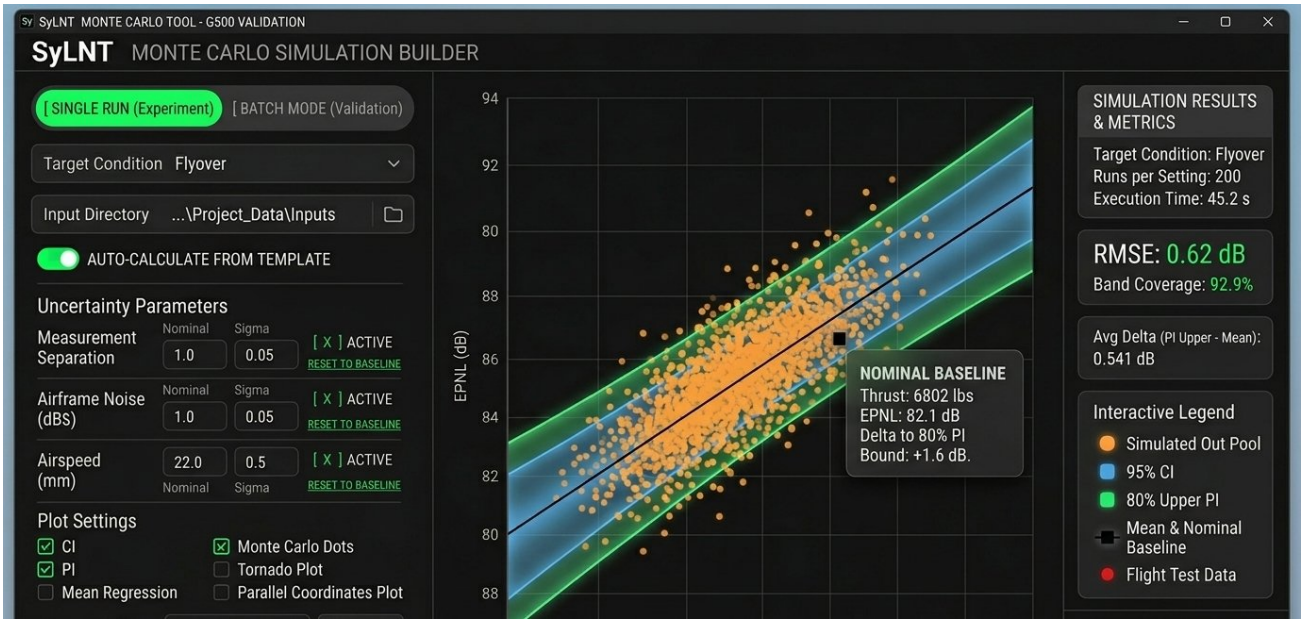
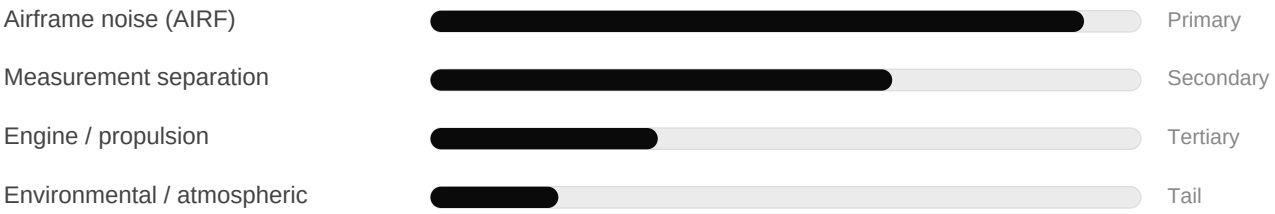


Fig. 3 — SyLNT App Designer interface: uncertainty parameter controls (left), EPNL Monte Carlo scatter with 80% PI band (centre), simulation metrics (right)

SENSITIVITY ANALYSIS

What drove the variance

Tornado sensitivity charts ranked all 8 input parameters by their contribution to the final EPNL envelope width across all 5 flight conditions. The hierarchy was consistent:



Airframe noise came out on top because the engine was specifically retargeted to isolate the AIRF array component directly, replacing the prior generic AFBB grouping. That change meant AIRF fluctuations fed straight through the logarithmic EPNL solver with nothing absorbing them. Environmental factors showed up mostly in the tails of the distribution rather than driving the core width of the 80% envelope.

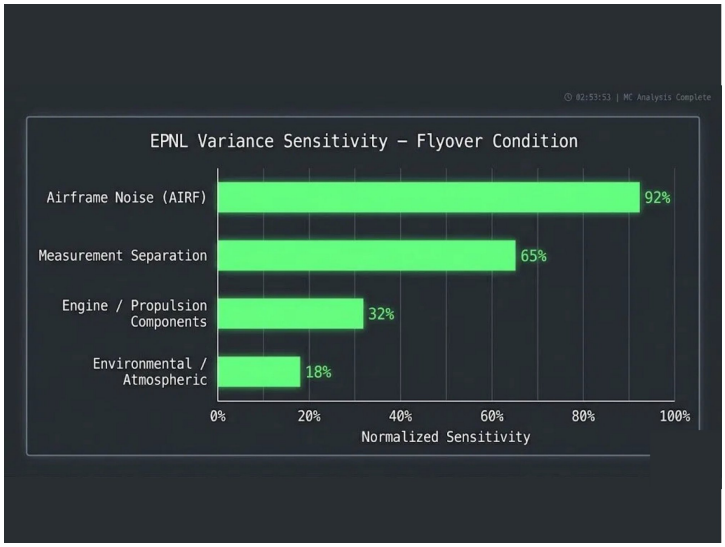


Fig. 4 — EPNL Variance Sensitivity Tornado chart — Flyover condition. Airframe noise (AIRF) dominates at 92% normalized sensitivity.

RESULTS

What the simulation confirmed

The simulation was run against MFA flight test data. By computing the difference between the upper prediction interval line and the mean regression, the results confirmed that the sigma values inherited from RTX Hartford were statistically consistent with P&WC; data. This gave the aeroacoustics team an independently validated uncertainty baseline for the first time.

On the performance side, switching to a parallelized architecture brought a full 5-condition validation sweep from 4.5 to 5 hours down to 25 to 30 minutes, roughly a 10x reduction. Engineers could now run the tool during a working session and act on the results the same day. The tool was handed off fully operational at the end of the co-op term and picked up by the team.

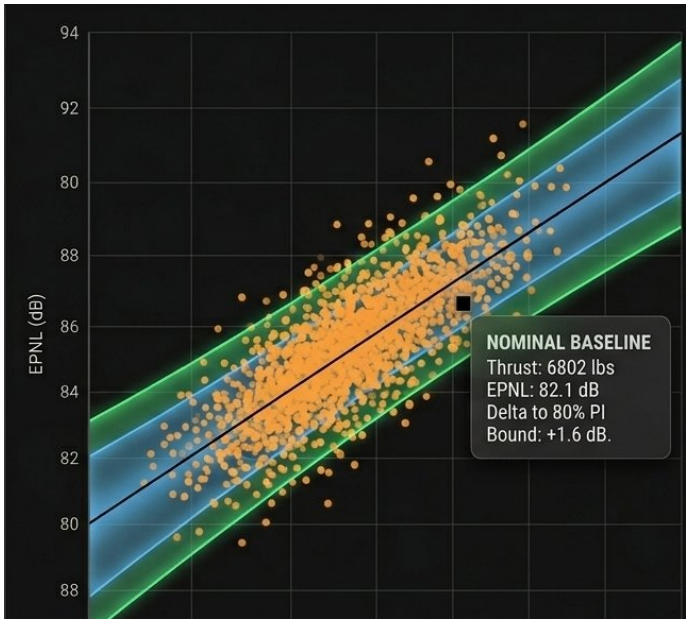


Fig. 5 — EPNL Monte Carlo scatter plot. Orange dots: simulated output pool. Blue band: 95% CI. Green band: 80% PI. Black line: mean regression. Nominal baseline at Thrust 6802 lbs, EPNL 82.1 dB.

RETROSPECTIVE

What I'd do differently

01 **ETL pipeline before the simulation loop**

The YS parser maps legacy PWA data formats and injects flight states while the simulation is already running, which adds overhead at the worst time. A proper pre-processing step that converts all inputs to a binary format like HDF5 or Parquet before the Monte Carlo loop starts would cut that out entirely and make each iteration faster.

02 **Port the math core out of MATLAB**

App Designer made the UI fast to build and parfor scaled well enough for this use case, but MATLAB carries licensing costs and heavier memory overhead per worker than is ideal. If I rebuilt this, the core aeroacoustic math would go into Python using NumPy and Numba or CuPy for GPU acceleration. The UI would be a separate web layer using React and Plotly, which would also make the interactive cursor functionality easier to extend.

03 **Automated correlation check on startup**

The sampling bug was found by looking at the output data and noticing the variance was too narrow. That is a fragile way to catch it. A short diagnostic run at startup (N=100) that computes a correlation matrix across all input variables would flag the problem directly. If nominally independent parameters like AIRF and Measurement Separation show R not equal to 0, the run stops before any validation data gets written.

REFERENCE

Technical specifications

Organization	Pratt & Whitney Canada (RTX)
Role	Lead software developer
Supervisor	Jong Park — Lead Engineer, Aeroacoustics
Stack	MATLAB, App Designer, Parallel Computing Toolbox
Simulation runs	10,000 iterations × 5 flight conditions = 50,000 per sweep
Input parameters	8 (incl. AIRF, measurement separation, engine components, atmospheric)
Convergence threshold	< 0.05 dB delta on 10th/90th percentile over successive blocks
Uncertainty envelope	80% confidence band (10th–90th percentile, non-parametric)
Parallelization	MATLAB parfor, 8–12 workers, inner-loop strategy
Runtime (serial)	4.5–5 hours per full sweep
Runtime (parallel)	25–30 minutes per full sweep (~10× speedup)
Regulatory baseline	ICAO noise certification standards
Outcome	Validated RTX Hartford sigma values against MFA data; tool handed off fully operational
Confidentiality	Public technical summary only — proprietary model, datasets, and validation methods excluded