https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Optimizing Gaussian Process Emulation and Generalized Additive Model Fitting for Rapid, Reproducible Earth System Model Analysis

Kunal Ghosh[1] and Leighton A. Regayre[1,2,3]

[1]School of Earth and Environment, University of Leeds, Leeds, LS2 9JT, UK
[2]Met Office Hadley Centre, Exeter, Fitzroy Road, Exeter, Devon, EX1 3PB, UK
[3]Centre for Environmental Modelling and Computation, University of Leeds, Leeds, LS2 9JT, UK

Correspondence: Kunal Ghosh (k.ghosh@leeds.ac.uk)

Abstract. Causes of model uncertainty in complex modeling systems can be identified using large perturbed-parameter ensembled (PPEs), combined with statistical emulators to increase sample size and enable variance-based sensitivity analyses and observational constraint. In global climate models such as the UK Earth System Model (UKESM), these approaches are typically applied at the global or regional mean scales for a limited set of variables. To accelerate progress in understanding the multi-faceted causes of climate model uncertainty, requires implementing such workflows at the model grid box scale, to enable analyses across variables that reveal how uncertainties propagate and interact spatially. However, this approach requires training millions of Gaussian process (GP) emulators and fitting an equal number of generalized additive models (GAMs) - a major computational bottleneck. We present a high-performance, open-source pipeline that introduces optimisations for this workflow. For GP emulation, we implement task-level parallelism and streamlined data handling on high-performance computing systems. For GAM fitting, we integrate a parallelized `pyGAM` interface with R's `mgcv::bam()` back end, using fast fREML estimation with discrete smoothing, memory-efficient batching, and improved input–output routines. These changes reduce GP training time by 97.5% (6177 → 154 s) and GAM fitting time by 95.2% (10623 → 511 s), yielding a ∼25× faster end-to-end workflow (96% total runtime reduction) and cutting peak memory use by a factor of 12. Outputs are numerically identical to the baseline implementation (Pearson correlation = 1.00 for both GP and GAM predictions). We demonstrate the approach using a UKESM PPE comprising 221 members scaled up to 1 million using GP emulators, and GAM fits applied to output for a single target variable, to show that the improved performance enables multi-variable, higher-resolution, and potentially multi-model analyses that were previously impractical. These improvements pave the way for PPE studies to scale in scope without compromising statistical fidelity, enabling more comprehensive exploration of model parameter uncertainty within feasible HPC budgets.

1 Introduction

Perturbed-parameter ensemble (PPE) studies are increasingly recognised as a vital tool in climate and Earth system modelling because they provide information needed to characterize model uncertainty and identify its causes. Well designed PPEs enable

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

systematic identification of structural model deficiencies, guide targeted model developments, and provide a basis for rigorous statistical calibration against observations (Carslaw et al., 2025), unlike multi-model ensembles (Stouffer et al., 2017; Durack et al., 2025), which conflate structural and parametric uncertainties in uncontrolled ways. Community assessments consistently argue that PPEs should be prioritised alongside increases in model complexity and resolution, as they offer one of the best opportunities to improve model reliability and provide robust uncertainty quantification for downstream impacts and decision making (Knutti, 2008; Carslaw et al., 2025). However, PPEs of complex models are computationally expensive, requiring hundreds to thousands of model runs. Additional computational costs are incurred during analysis by expanding the ensemble size to millions of parameter combinations using an appropriate form of model emulation and by statistically decomposing the variance associated with the millions of combinations. Recent progress harnessing the power of climate model PPEs have utilized statistical emulators, such as Gaussian Process (GP) surrogates (Oakley and O'hagan, 2002; O'Hagan, 2006), which enable fast approximations of expensive model components, and flexible smoothers such as generalized additive models (GAMs) (Wood, 2020; Servén et al., 2018) to decompose variance, identify dominant sources of uncertainty, and provide interpretable functional relationships (e.g. Regayre et al., 2025; Prévost et al., 2025). Ideally, these techniques would be applied at existing model temporal and spatial resolution. However, the lack of scalability in creating GP emulators, sampling millions of parameter combinations from each emulator, then implementing and evaluating GAMs severely limit the scope of climate model PPE analyses.

The major computational bottleneck to widespread implementation of statistical emulation of PPEs and subsequent variance decomposition are caused by the computation costs incurred in evaluating multiple model variables within each model grid box - a necessary step to establish interpretable functional relationships between variables. For each model variable, implementation of these methods at moderate temporal and spatial resolution requires training millions of independent GP emulators and GAM analyses, which requires tens of hours of runtime and hundreds of gigabytes of memory (Yoshioka et al., 2019; Johnson et al., 2015; Lee et al., 2011; Johnson et al., 2018). As a result, prior applications have often been restricted to single variables (Regayre et al., 2025), individual months or seasons (Prévost et al., 2025), or coarse regional aggregation (Regayre et al., 2014; Johnson et al., 2020; Regayre et al., 2023). This limited analysis scope hinders efforts to maximally exploit PPEs for multi-variable, multi-region and multi-model studies.

Several frameworks have advanced the practical use of statistical emulators in climate science. Watson-Parris et al. (2021) introduced the Earth System Emulator (ESEm), a general framework for building GP surrogates across a range of model components, while Yang et al. (2025) proposed an additive GP approach to improve interpretability of parameter–output relationships. Multi-scale GP methods (Susiluoto et al., 2020) and multi-fidelity emulators (Servera et al., 2023) have also addressed scaling challenges for remote sensing and radiative-transfer models. However, these approaches rely on statistical approximations or structural modifications of the emulator, which may reduce accuracy or reproducibility compared to directly leveraging high-performance computing resources. Similar challenges have been reported for scaling GAMs: Wood et al. (2015) showed that fitting spatiotemporal GAMs to gigadata from the U.K. Black Smoke monitoring network (almost 10 million daily observations) was computationally prohibitive using conventional methods, with model matrices alone requiring hundreds of gigabytes of memory. Their solution combined discretisation of covariates with parallelised matrix operations to

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

achieve order-of-magnitude speed-ups, reducing runtimes from months to hours, though at the cost of some flexibility and potential loss of fine-scale precision. Together, these examples underline both the importance and the difficulty of deploying GP and GAM methods at scale, and highlight the urgent need for further innovation when such approaches are applied within PPE workflows involving millions of model evaluations.

In this study, we introduce an open-source pipeline designed to remove the computational bottlenecks of GP emulation and GAM fitting in PPE workflows. For the GP emulation stage, we implement task-level parallelism and streamlined data handling to enable efficient use of high-performance computing resources (e.g. Lawrence et al., 2013). For the GAM stage, we likewise employ task-level parallelism together with R's `mgcv::bam()` function (fast fREML estimation with `discrete=TRUE`), hereafter "rBAM," which combines discrete smoothing, batching, and optimised input–output operations. Together, these enhancements accelerate the combined workflow by a factor of about 25 (a 96 % reduction in total runtime, from ∼16,800 to 665 s). Additionally, the enhancements reduce peak memory demand by more than an order of magnitude, while reproducing baseline outputs to a high degree of numerical precision. We demonstrate performance gains using a single-variable example from a climate model PPE comprising 221 simulations. We analyse 833 grid boxes per month across the European domain. In each grid box, we train a GP emulator on the 221-member PPE, then evaluate 1,000,000 parameter combinations via the emulator to generate the extended ensemble, and fit a GAM to those 1,000,000 outputs for that grid box. This is done independently for each month (833 grid boxes × 12 months = 9,996 emulators and 9,996 GAM fits). Note, for ease of interpretation of our approach we include a lexicon of high-performance and statistical terms in Appendix A. By removing long-standing scaling limitations, the pipeline enables broader application of PPE methods such as detecting structural model deficiencies and constraining parametric uncertainty, making high-resolution, multi-variable, and even multi-model analyses tractable. Although demonstrated here for climate model PPEs, the pipeline is broadly applicable to other large-scale emulation and regression problems where statistical fidelity and computational efficiency are critical.

2 Exemplar problem

To demonstrate and benchmark our approach, we focus on a PPE generated with the UK Earth System Model version 1 UKESM1-A; Sellar et al. (2019)), as developed and described in Regayre et al. (2023). This ensemble is a suitable exemplar because it combines scientific relevance, the PPE was designed to constrain aerosol–cloud interaction radiative forcing uncertainty - one of the largest sources of climate model uncertainty (Bellouin et al., 2020), with the computational challenges of handling very high-dimensional model output. The dataset comprises 37 perturbed parameter combinations within 221 PPE members expanded by statistical GP emulation to one million parameter combinations in each model grid box. GAMs are subsequently fitted to decompose variance and assess parameter sensitivities, again at the model grid box scale. In total, to quantify causes of aerosol–cloud interaction uncertainty across 833 grid boxes per month (9,996 grid boxes for 12 months), the workflow involves training one GP emulator for each grid box on the 221-member PPE, sampling one million parameter combinations from each emulator, and fitting a GAM to those emulated outputs. This configuration is therefore representative of the scale required for multi-variable, multi-regional, or multi-model studies.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

This configuration stresses both stages of the workflow. GP training must handle high-dimensional input space and repeated tasks at scale, while GAM fitting must operate efficiently on high-resolution spatiotemporal fields without resorting to coarse aggregation. Initial baseline implementations of these steps proved computationally prohibitive on HPC systems, motivating the optimisations described in this paper.

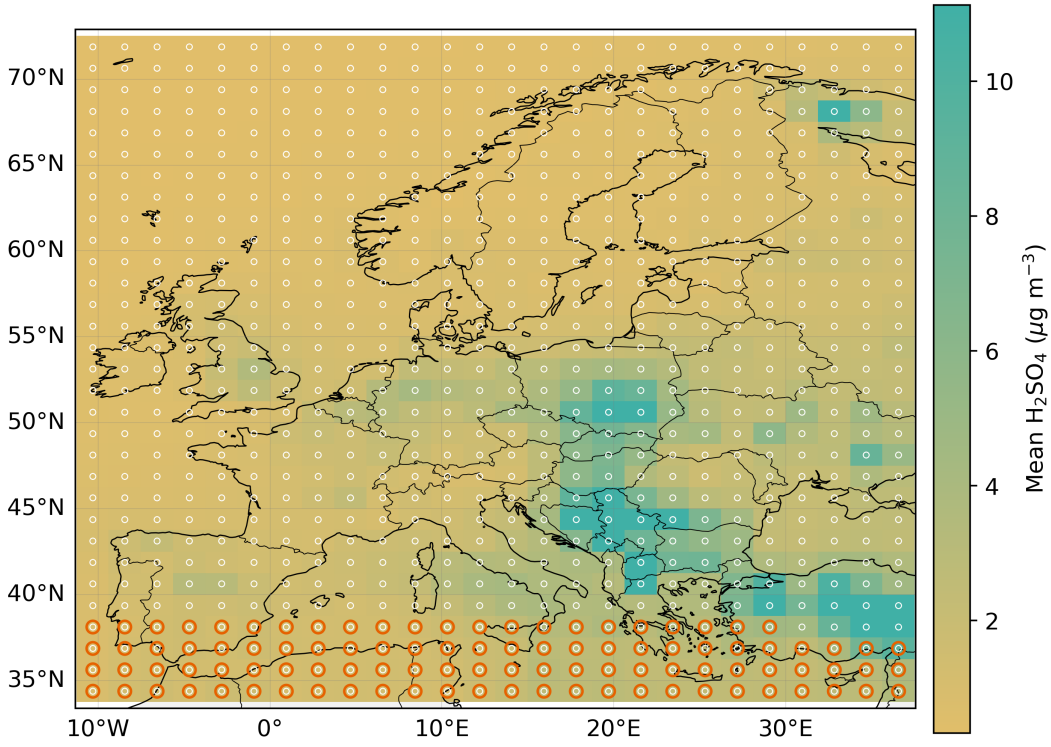

Figure 1. UKESM1 PPE domain using one example target variable: mean H_2SO_4 concentration (μg m^{-3}) for January 2017. The map shows the European domain with grid points centers marked by white circles. Orange circles highlight the first 100 grid points selected for in-depth analysis in this work.

Figure 1 shows the UKESM1 PPE domain using one example target variable, the monthly mean H_2SO_4 concentration for January 2017. This illustrates both the spatial coverage of the European domain and the distribution of grid points required for emulator training and variance analysis. A subset of 100 model grid boxes was used to validate workflow performance before scaling to the full set of 9,996 grid points for the optimised workflow.

2.1 Benchmarking setup

All performance tests were carried out on the JASMIN "LOTUS" high-performance computing (HPC) cluster hosted at the UK Centre for Environmental Data Analysis (CEDA) (Lawrence et al., 2013). The LOTUS cluster is a heterogeneous system with several thousand CPU cores available across multiple partitions. For this study, we used standard CPU nodes, each

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

equipped with dual Intel Xeon processors (2.1–2.6 GHz) and 192 GB RAM, interconnected by high-speed InfiniBand. Jobs were scheduled using Slurm (v22.05), and all experiments were submitted through Slurm job arrays with task-level resource allocation. Unless otherwise noted, each task was executed on a single CPU core with 10–16 GB of memory, while concurrency was controlled using the `#SBATCH -array` directive with throttling set between 200 and 500 simultaneous tasks to minimise filesystem contention.

Original dataset sizes mirrored those used in the scientific analysis such as those reported in Regayre et al. (2025, 2023); Lee et al. (2012, 2016); Johnson et al. (2020), though analyses were restricted in those studies. For benchmarking purposes, we measured end-to-end walltime, peak memory usage, and accuracy equivalence across both the currently used (baseline) and newly developed (optimised) workflows. Accuracy was quantified using Pearson correlation and root mean squared error (RMSE) between baseline and optimised outputs, while scalability was assessed as a function of task count and concurrency level. Default walltime limits were set to 24 h for all experiments, with memory limits varied between 10 and 32 GB per task depending on workload.

All software were used in stable production releases available at the time of analysis. Python (v3.10) was used for pipeline orchestration, plotting, and baseline GAM fitting with `pyGAM` (v0.9.0)(Servén et al., 2018). R (v4.2.2) (R Core Team, 2022) was used for GP emulation with `DiceKriging` (v1.6.0) (Roustant et al., 2012) and for optimised GAM fitting via `mgcv` (v1.8-41)(Wood et al., 2015). Additional R packages included `readr` (Wickham and Bryan, 2023), `lhs` (Carnell, 2022), `sensitivity` (Pujol et al., 2017), `trapezoid` (Sottile and Gohel, 2022), and `truncnorm` (Mersmann, 2022). Parallelisation was supported by Slurm job arrays on the cluster side, and OpenMP (v4.5) within `mgcv::bam()` for multithreaded GAM fitting.

2.2 Baseline workflow and bottlenecks

2.2.1 Baseline GP emulation.

The baseline workflow combined GP emulation and GAM fitting using widely available Python libraries. GP emulators were trained with the `scikit-learn` `GaussianProcessRegressor` using a squared-exponential kernel with automatic relevance determination. Training was performed sequentially, with one emulator fitted per parameter setting, and relied on dense kernel inversions that scale cubically with the number of training points. Although individual emulators were modest in size, the aggregate cost of fitting tens to hundreds of thousands of models became prohibitive. Intermediate emulator outputs were written to disk for subsequent analysis, introducing additional input/output (I/O) overhead.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

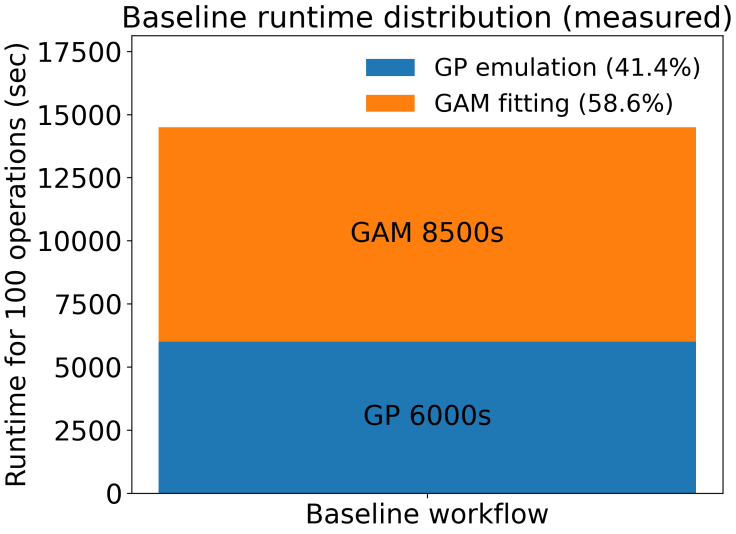

Figure 2. Measured runtime distribution of the baseline workflow over 100 operations.

2.2.2 Baseline GAM fitting.

GAM fitting was carried out entirely in Python using the `pyGAM` library with cubic regression splines. Fits were implemented serially at the grid box level, with each requiring repeated construction of large design matrices and independent smoothing-parameter optimisation. This approach was straightforward to implement and completely reproducible, but computationally inefficient as no batching or parallelism was possible, and memory was not shared across tasks.

Profiling of this baseline workflow, summarised in Fig. 2, showed that GAM fitting using one million combinations of 37 parameter dimensions dominated total runtime and was also the main contributor to peak memory use, which regularly exceeded 80 GB during spline matrix assembly. Across 100 operations, GAM fitting required ∼8,500 s (58.6% of total wall time) and GP emulation ∼6,000 s (41.4%), giving a total of ∼14,500 s. Disk I/O overhead was negligible at this scale but becomes significant at larger ensemble sizes due to the churn of intermediate files.

2.2.3 Baseline (median-hold) variance computation.

In the baseline implementation, the marginal importance of each parameter was obtained by a "median-hold" procedure. For each parameter x_i, the GAM model predictions were recomputed while varying x_i across the extended ensemble and holding all other parameters fixed at the median values used to train the GAM. The variance of the resulting one-dimensional prediction series quantifies variable response to changes in x_i. Repeating this process for all 37 parameters yields a set of per-parameter variances $\mathrm{Var}(f_i)$ and gradient signs $\mathrm{sign}[\mathrm{Cov}(x_i, f_i)]$ that can be compared to quantify the relative importance of model parameters as causes of variance. Although this approach is conceptually straightforward, it requires rebuilding the prediction matrix P times and is therefore computationally expensive.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

The baseline workflow had three bottlenecks. First, GP emulation ran as a loop-based wrapper with dense-kernel inversions executed sequentially, leaving the embarrassingly parallel PPE analysis structure under-utilised. Second, GAM fitting in `pyGAM` was serial, using cubic splines with cross-validation smoothing, which scaled poorly with the number of grid boxes and caused repeated disk access. Third, the stages communicated via per-grid *text* `.dat` files (i.e., many tiny text artefacts), creating tens of thousands of small files and heavy metadata traffic on the shared file system. On the JASMIN super-data-cluster (Lawrence et al., 2013), where jobs are limited to 36 h and 64–128 GB per node, these bottlenecks led to 20–30 h runtimes for a single large-scale experiment and frequent out-of-memory failures.

3 Optimised workflow

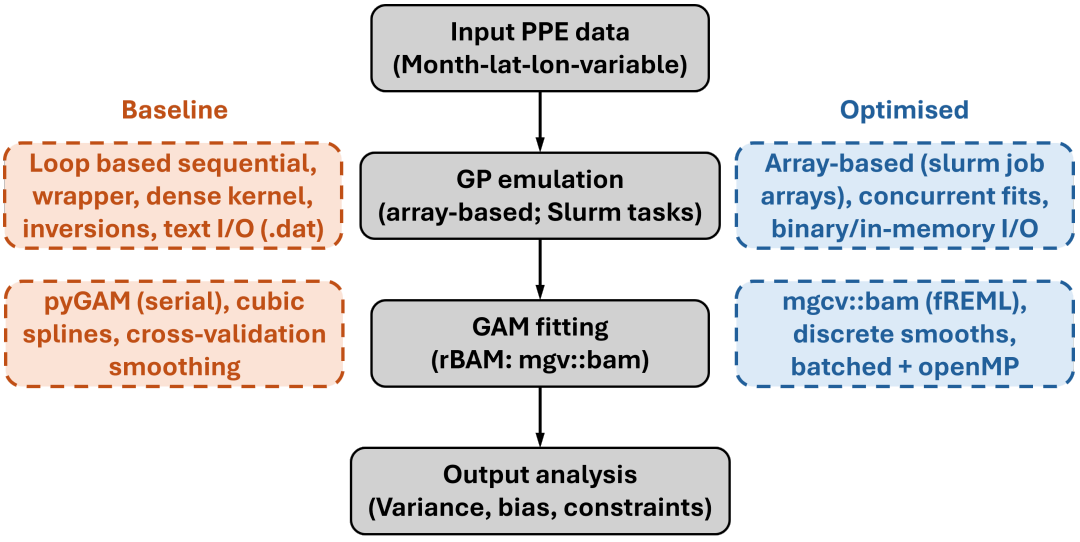

Figure 3. High-performance statistical emulation pipeline. Grey boxes show the main workflow stages; orange boxes indicate baseline implementations; blue boxes indicate the optimised implementations. The GP emulation stage trains a Gaussian Process emulator for each grid box and generates one million parameter samples per emulator. The GAM fitting stage applies generalised additive models to the emulated outputs to decompose variance and identify parameter sensitivities.

The optimised pipeline targets both dominant costs identified in the baseline workflow: GP emulation and GAM fitting. The design principle is to preserve the statistical formulation while restructuring execution for high concurrency, bounded memory, and minimal I/O. Across the full workflow, these changes yield an overall speed-up of $\sim 25\times$ (96 % total runtime reduction) and reduce peak memory by a factor of 12, with outputs matching the baseline to numerical tolerance. Figure 3 summarises the end-to-end process; implementation details are given below and a baseline–optimised comparison is provided in Table 1.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

3.1 Optimised Gaussian process emulation stage

In the baseline workflow, GP surrogates were trained and evaluated via serial loops that spawned a small number of competing `Rscript` processes on a single node, wrote large text intermediates, and suffered from straggler tasks. The optimised implementation retains the same emulator (`DiceKriging::km`) and large-sample predictor but replaces the control flow with a task table and Slurm arrays: each (month, latitude, longitude) combination maps to a single array task. This enables cluster-wide concurrency, deterministic task assignment, and isolation of failures for simple re-runs as needed. Per-task memory is bounded explicitly, and prediction output is written in binary or kept in memory to avoid filesystem churn.

Figure 4 compares probability density functions (PDFs) of mean H_2SO_4 for January 2017 at four randomly selected grid points, derived from one million parameter combinations. Baseline (orange) and optimised (purple) curves are indistinguishable, and Pearson correlation between implementations equals 1.000, demonstrating that the optimised workflow preserves statistical fidelity.

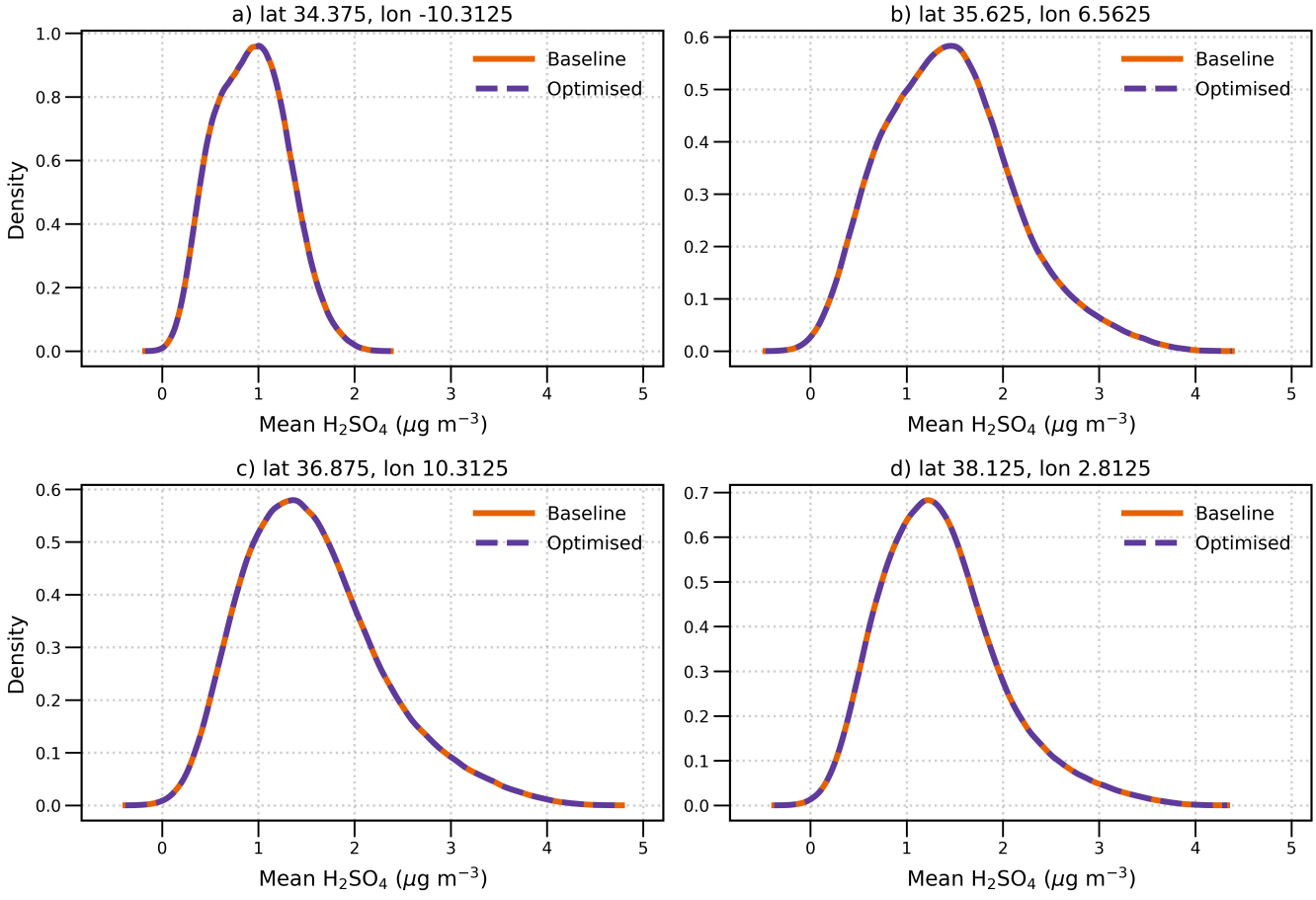

Figure 4. Probability density functions of mean H_2SO_4 for January 2017, obtained from one million parameter combinations using GP emulators. Panels (a–d) show results for four randomly selected grid points, identified by their latitude and longitude.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Performance scaling is shown in Fig. 5. In the baseline workflow, runtime increases steeply with task count, exceeding 6,000 s for GP emulation in 100 model grid boxes and becoming superlinear owing to node-level contention and a loop-based batch wrapper that executes emulators sequentially within each batch (i.e. not true parallelism). In the optimised workflow, we launch one emulator per Slurm array task and utilise up to **200 CPU cores** concurrently, achieving genuine task-level parallelism. Wall time remains nearly constant over three orders of magnitude in task count, and the 8,000-task case completes ∼**40**× faster than the baseline. This refactoring converts the GP stage from a sequential bottleneck into a scalable, cluster-parallel process.

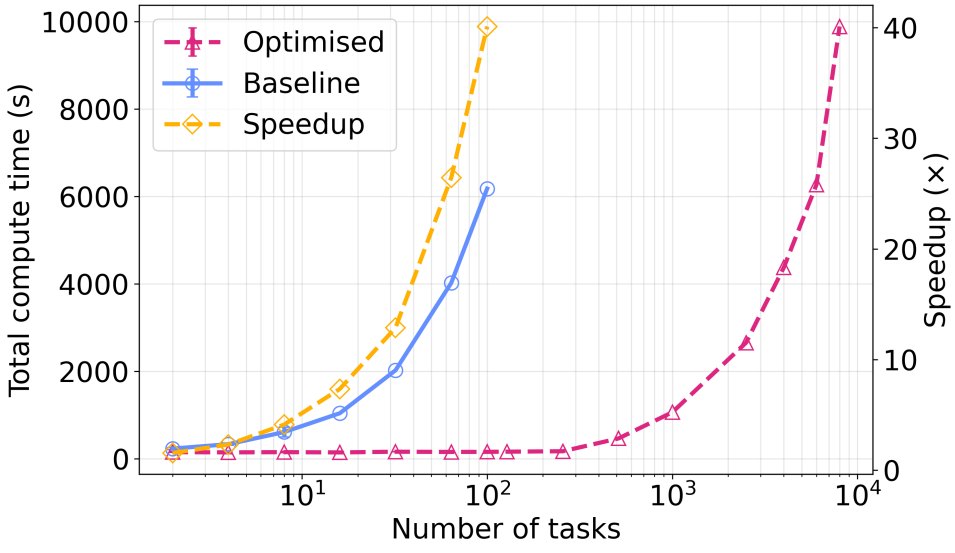

Figure 5. Scaling of the GP emulation stage. Total runtime for the baseline and optimised workflows (left axis), with relative speed-up (right axis).

3.2 Optimised GAM fitting stage

The baseline workflow used `pyGAM` to fit cubic regression splines serially at the grid-box level, rebuilding large design matrices and re-optimising smoothing parameters for each fit. The optimised approach employs an R-based Big Additive Model (hereafter rBAM) using `mgcv::bam()`, leveraging fast restricted maximum likelihood (fREML) estimation with `discrete=TRUE` for large datasets, OpenMP multithreading within fits, and Slurm array execution across tasks. Filtered design matrices are prepared once per task, intermediate reads are minimised, and outputs are written deterministically. This reduces both runtime and peak memory per fit while leaving the statistical model unchanged.

3.2.1 Vectorised ”terms-based” variance and sign computation

After fitting the `bam` model we obtain all parameter contributions to variance simultaneously using `predict(..., type="terms")`, which returns a matrix $\mathbf{T} = [t_1, \ldots, t_P]$ with one column per parameter, with the GAM evaluated at all N rows. We define each

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

parameter's variance contribution as $\mathrm{Var}(t_i)$ and its effect sign as $\mathrm{sign}(\mathrm{Cov}(x_i, t_i))$. These quantities are invariant to the additive constants that `type="terms"` may include for each evaluation, since $\mathrm{Var}(t_i + c) = \mathrm{Var}(t_i)$ and $\mathrm{Cov}(x_i, t_i + c) = \mathrm{Cov}(x_i, t_i)$. This vectorised approach is numerically equivalent to the baseline "median-hold" evaluation while avoiding repeated design-matrix builds and `predict` calls: it computes all P partial effects in one batched pass and then discretises using simple column-wise statistics. Using this "terms-based reduction" collapses P separate predictions into one batched call (with `block.size` tuned to memory), which—together with `bam(..., discrete=TRUE)`—accounts for much of the $\geq 20\times$ speed-up observed in the optimised GAM stage while preserving numerical fidelity (see Fig. 7).

3.2.2 Interpretation of the variance term.

The variance term $\mathrm{Var}(t_i)$ quantifies the marginal contribution of parameter x_i to the modelled response variance, given the joint additive model $\eta = \sum_{i=1}^{37} f_i(x_i)$. Each smooth fit $f_i(x_i)$ is estimated simultaneously with all others, so $\mathrm{Var}(t_i)$ reflects the variability in the partial fit after accounting for the influence of the remaining parameters. It therefore measures the *marginal importance* of x_i rather than the total variance that would be obtained if parameter x_i were perturbed in isolation. Fitting a single-parameter GAM would attribute all variance to that parameter and yield a much larger, non-comparable value. The sum of all $\mathrm{Var}(t_i)$ is not equal to $\mathrm{Var}(y)$ because the smooths may covary:

$$\mathrm{Var}\left(\sum_i f_i(x_i)\right) = \sum_i \mathrm{Var}(f_i) + 2\sum_{i<j} \mathrm{Cov}(f_i, f_j).$$

Consequently, the reported variances represent the marginal sensitivity of the model output to each parameter within the full 37-parameter framework, consistent with the baseline median-hold method (Figs. 6–B1).

We built a faster way to calculate how each model parameter affects the GAM output. Instead of recomputing predictions one by one, we get all parameter effects in a single call and compute their variances and signs. The variance of each effect shows how much that parameter influences the output, after accounting for all other parameters not by itself, but as part of the full 37-parameter system.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

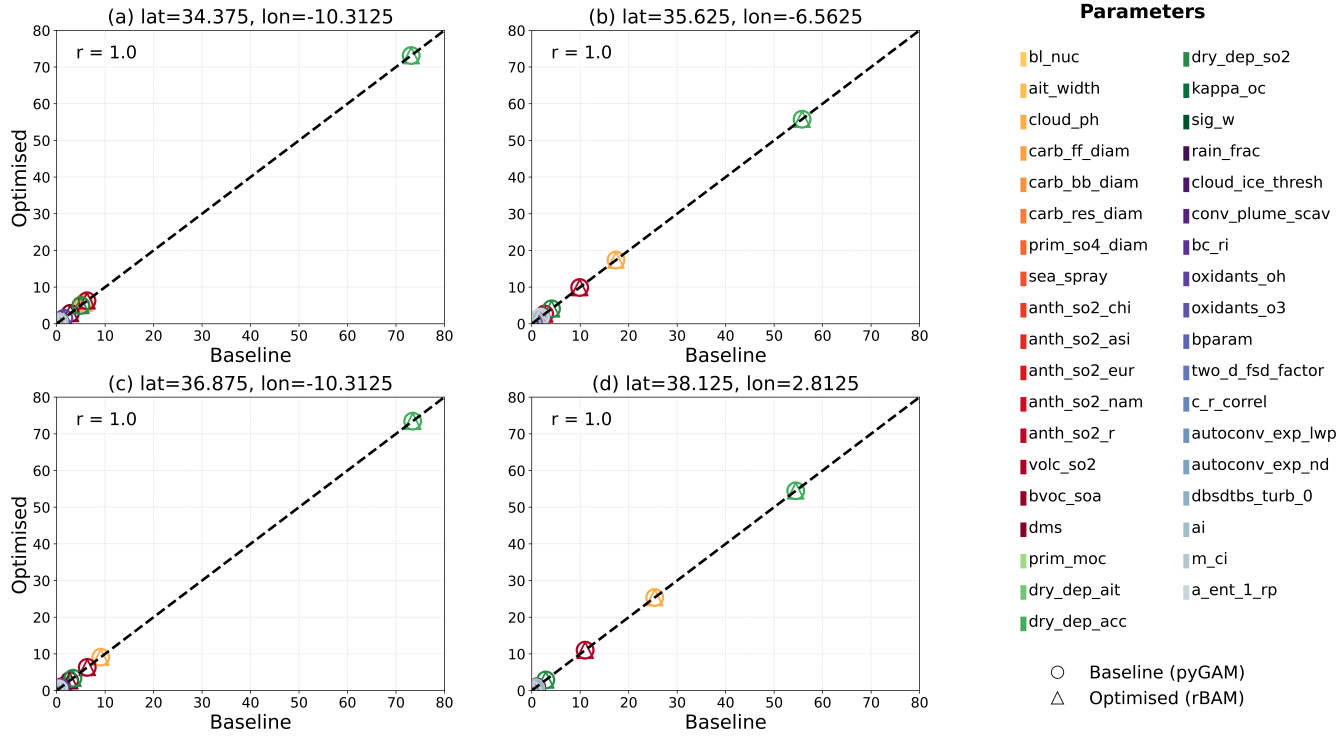

Figure 6. Variance contributions from GAM fitting for four grid points. Baseline (`pyGAM`, x-axis) versus optimised rBAM (y-axis).

Figure 6 compares fractional variance analyses from baseline and optimised workflows. The points fall exactly on the 1:1 line, with Pearson correlation $r = 1.0$, showing that rBAM reproduces the variance decomposition obtained with `pyGAM`. Detailed per-parameter variance tables for four different grid cells are provided in Appendix B (Tables A1–A4).

The efficiency gains are shown in Fig. 7. In the baseline workflow, runtimes increased superlinearly with the number of tasks, exceeding practical limits beyond GAM fits in a few hundred model grid boxes, while peak memory often surpassed node capacity. In the optimised workflow, rBAM (`mgcv::bam` with fREML and discrete smoothing fits) maintains near-constant wall time over three orders of magnitude in task count, achieves more than $20\times$ speed-up at 1,000 tasks, and reduces peak memory by over an order of magnitude. Crucially, rBAM's much smaller memory footprint relative to `pyGAM` allows one GAM fit (model grid box) per Slurm array task and *full utilisation* of **200 CPU cores** in parallel—true task-level parallelism that delivers the real throughput boost, mirroring the optimised GP emulation stage. This performance enables variance decomposition at global scale with fine spatial resolution and across multiple variables, extending the scope of PPE analysis beyond what was previously feasible.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

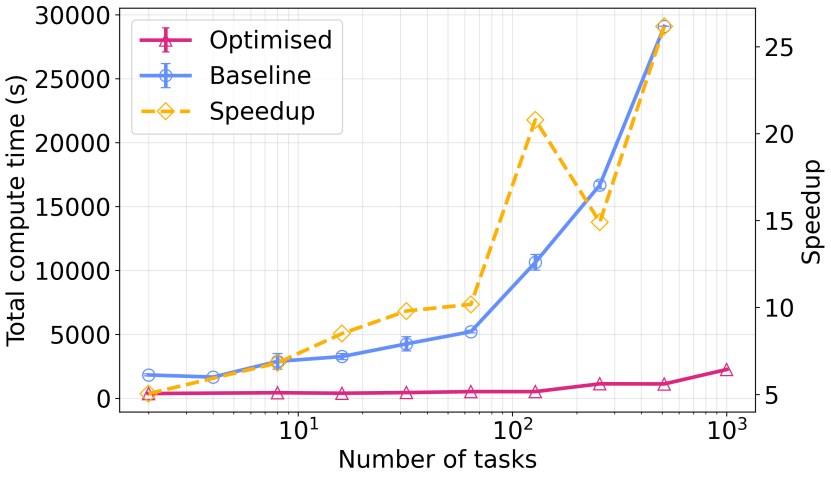

Figure 7. Scaling of the GAM fitting stage. Total runtime for the baseline and optimised (left axis), with relative speed-up (right axis).

3.3 Pipeline integration and reproducibility

The pipeline implements and automates the workflow for large-scale statistical emulation. In this context, the workflow refers to the ordered sequence of tasks required to generate, emulate, and analyse model output, specifically, GP emulation, GAM fitting, and downstream analysis. The pipeline, by contrast, is the software and organisational framework that executes this workflow reproducibly and efficiently on high-performance computing systems. It manages task scheduling, data handling, and parallel execution, ensuring that the workflow can be scaled and repeated without manual intervention.

The pipeline organises the optimised workflow around a task-centred folder layout with separate stages for GP emulation, GAM fitting, and downstream analysis. Configuration is provided via plain-text (.YAML/.JSON) files (month, grid, variable), enabling deterministic re-runs that reproduce identical results from the same inputs. A consistent file-naming scheme based on month, latitude and longitude indices supports automatic discovery of missing or failed tasks. In previous implementations, a single task failure, often caused by node timeouts or memory overuse, could halt or invalidate large batch jobs, requiring manual log inspection and reruns of the entire workflow. In the optimised pipeline, each task writes an individual log and output file identified by its coordinates, allowing a lightweight post-check script to detect missing or incomplete outputs and automatically resubmit only the affected tasks. This design prevents error propagation, eliminates manual recovery steps, and greatly improves overall robustness and throughput on HPC systems. Slurm job arrays orchestrate parallel execution for a given output variable, with each array index corresponding to a single model grid box for one month (one task). The array structure can span any number of grid boxes, regional or global, depending on available HPC resources. Inputs are read once per task; outputs are kept in memory or written in binary to limit I/O overhead. Per-task logs (`slurm_output_%A_%a.out`) record program outputs and error messages, and a lightweight post-check script scans for failures and automatically resubmits only the affected tasks.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Software environments are fully version-controlled to ensure reproducibility. All Python dependencies are specified in the Conda environment file `envs/environment.yml`, while all R packages (including `mgcv` and `DiceKriging`) are declared in the manifest script `envs/install_R_deps.R`. This script automatically installs the required packages from CRAN and records the complete R session information (`sessionInfo()`), providing an explicit record of the package versions used in the analysis. The workflow is designed to run efficiently on high-performance computing (HPC) systems (e.g., JASMIN, ARCHER) and can also be executed locally on Windows, Linux, or macOS. The repository includes a fully self-contained demonstration using four grid points (H_2SO_4, January 2017) that allows users to test the complete GP–GAM pipeline without requiring HPC access. Software environments are fully reproducible using the provided Conda specification (`envs/environment.yml`) for Python and the manifest script (`envs/install_R_deps.R`) for R.

The individual improvements to GP emulation, GAM fitting, and workflow integration are brought together in Table 1, which contrasts key features of the baseline and optimised workflows.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Table 1. Baseline versus optimised workflows at the GP and GAM stages (key changes highlighted).

Aspect	Baseline	Optimised
GP emulation stage		
Software	R `DiceKriging` (`km`/`predict.km`) via looped scripts	Same emulator/predictor, orchestrated with **Slurm arrays**
Algorithm	Serial training; large-sample prediction; text intermediates	Same algorithms; **task-table orchestration** and **binary/in-memory I/O**
Parallelisation	Few competing `Rscript` jobs on one node	**Cluster-wide concurrency**; one task per (month, lat, lon)
Memory	Multiple processes share node RAM; high peaks	**Bounded per-task RAM**; minimal footprint
I/O	Shared/interleaved logs; many small files	**Per-task logs**; deterministic filenames
GAM fitting stage		
Software	Python `pyGAM`	Python–R bridge to **`mgcv::bam()`** (rBAM)
Algorithm	Cubic splines; CV-based smoothing; rebuild design matrices per fit	**fREML** with `discrete=TRUE`; **batched** design matrices
Parallelisation	Serial fits; limited threading	**OpenMP within fits + Slurm arrays** across tasks
Memory	Large, repeatedly built matrices; peaks >80 GB	**Sparse/batched** design; substantially lower peaks
I/O	Filenames without standardisation; repeated reads	**Deterministic outputs**; minimal re-reads

4 End-to-end pipeline performance

The optimised workflow delivers large and systematic improvements in both runtime and memory at every stage of the pipeline (Figure 8 and Table C1). Median per-task walltime for the GP emulation falls from 6,177 s to 154 s (a **40.1**× speed-up), while GAM fitting drops from 10,623 s to 511 s (**20.8**×). Taken together, the end-to-end time reduces from 16,800 s to 665 s—about **25.3**× faster overall (from ∼4 h 40 min to ∼11 min per model grid box).

Peak memory usage shows comparable gains. GAM fitting, which dominated the baseline memory footprint, decreases from ∼100 GB to ∼10 GB (**10.0**× reduction), and the GP emulation stage falls from ∼50 GB to ∼6.5 GB (**7.7**×). As a result, the end-to-end peak memory requirement contracts by around **10.0**×, enabling much denser job arrays on the same hardware and reducing the frequency of errors caused by memory limitations.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Practically, these improvements come from replacing slow, memory-intensive methods and I/O patterns with batched and streaming operations, using more efficient GAM fitting (e.g. discrete smoothing fits with `bam`) and lean GP emulation evaluation paths, while preserving the statistical specification of the original techniques. Figure 8 provides a side-by-side comparison of the GAM fitting, GP emulation, and total pipeline speed-ups to highlight where the largest gains are realised.

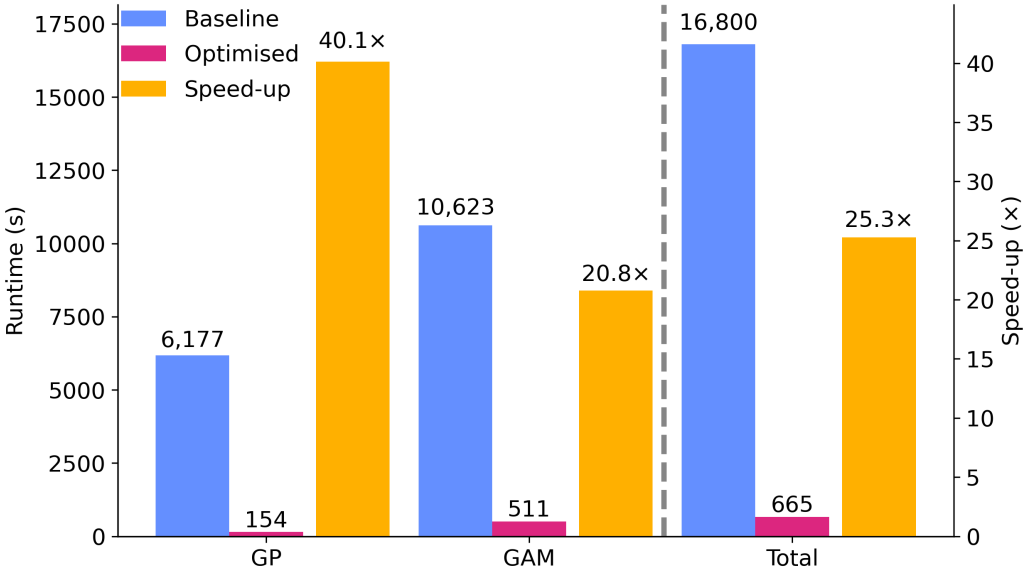

Figure 8. Bar chart of speed-up factors showing GAM fitting, GP emulation, and total pipeline side-by-side for quick comparison. Baseline and optimised runtimes (left axis) are shown alongside the speed-up (right axis). GP achieves the largest gain (40.1×; 6,177 s → 154 s), GAM is 20.8× faster (10,623 s → 511 s), and the full pipeline is reduced by 25.3× overall (16,800 s → 665 s).

5 Application potential

The optimised pipeline substantially expands what is feasible in PPE-based climate model analysis. By reducing end-to-end runtime by more than an order of magnitude and lowering memory demand twelve-fold, our new workflow opens the door to in-depth analyses that were previously unachievable due to computational bottlenecks. In particular, four scientific areas of exploration are now enabled: (i) simultaneous treatment of multiple target variables, such as aerosol optical depth, cloud droplet number, and liquid water path, rather than single-variable case studies; (ii) finer temporal resolution, moving from annual mean to monthly, daily or even sub-daily outputs; (iii) extension to larger spatial domains, including global analyses without the need for coarse aggregation of model data; and (iv) cross-model comparisons in which multiple PPEs can be emulated and analysed under a unified statistical framework.

Example application scenarios include quantifying shared caused of aerosol radiative forcing uncertainty across multiple climate models, constraining interacting sources of uncertainty across model components and evaluating constraints across

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

models to identify key structural model deficiencies in the current generation of climate models. The workflow is also readily applicable to non-climate large-ensemble settings where emulation and variance decomposition are required, such as hydrological forecasting, air-quality assessments or Earth system model evaluations.

Some opportunities for further refinement remain. We implemented discrete smoothing using rBAM, which significantly accelerates GAM fitting but may require parameter tuning when applied to extremely noisy or non-stationary predictors (not yet experienced in our chosen climate model outputs). Choice of kernel in GP emulator creation affects emulator fidelity - automated selection procedures could be implemented to trade some of the computation gains with improved emulator skill, which will be important as analyses scale. Finally, although per-task memory ceilings have been reduced substantially, very large multi-variable or multi-model applications, where distinct parameters are perturbed over model-specific ranges, will need more complex job-array design and, in some cases, access to distributed or GPU-enabled resources.

6 Conclusions

We have introduced and benchmarked a high-performance, open-source pipeline for large-scale statistical emulation in Earth system science. By restructuring workflow execution around task-level parallelism, streamlined I/O, and memory-efficient smoothing, we reduced GP training time by 97.5 % and GAM fitting time by 95.2 %, yielding a ∼25-fold end-to-end speed-up and a 12-fold reduction in peak memory demand. Crucially, these efficiency gains were achieved without loss of statistical fidelity, where emulator predictions and variance decompositions are numerically identical to the baseline implementation.

The workflow is designed for reproducibility and ease of adoption. All components are open-source, orchestrated via job arrays or containerised environments, and follow a deterministic file structure that simplifies reruns and failure recovery. This makes the pipeline directly reusable in other modelling contexts where emulators and smoothers are applied to high-dimensional outputs.

Future work will extend the framework in three directions: automatic kernel and smoothing-parameter selection to reduce the need for manual tuning, and distributed inference across heterogeneous HPC and cloud platforms. By combining statistical fidelity with computational tractability, the pipeline provides a scalable foundation for next-generation PPE studies and for broader applications of GP emulation and fractional variance decomposition methods in environmental science.

Code and data availability. All source code developed for this study, including pipeline scripts, configuration files, and figure-generation notebooks, is publicly available through the GitHub repository gp-gam-optimisation-pipeline. A tagged version corresponding to this paper is permanently archived on Zenodo (10.5281/zenodo.17543623). The repository includes a top-level `README.md` file providing step-by-step instructions and exact commands to reproduce all benchmarks, figures, and tables.

All simulation outputs required to reproduce the figures and tables in this paper are available as an accompanying dataset on Zenodo (10.5281/zenodo.17544324). This dataset includes emulator predictions, GAM fitting outputs, and benchmarking measurements for both baseline and optimised workflows, together with metadata describing input PPE configurations.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Appendix A: Lexicon of computational and workflow terms

This lexicon defines key computational and statistical terms used throughout the paper to aid readers who may be less familiar with high-performance computing and emulation workflow terminology.

Term / Acronym	Definition and Explanation
Accuracy equivalence	Verification that optimised and baseline outputs are statistically identical, confirming no (or minimal to a specified level of accuracy) scientific change.
Array directive (Slurm)	A scheduler instruction that defines how many tasks to launch in a job array and optionally limits concurrent execution, for example `#SBATCH -array=0-999%500`. Used to implement throttling between 200 and 500 simultaneous tasks to balance load and filesystem performance.
Automatic resubmission	Recovery process in which failed tasks are detected and rerun automatically without manual intervention.
Baseline workflow	The original serial implementation of the GP–GAM procedure that used sequential loops, text-based I/O, and no parallelisation. It provided reference results for benchmarking.
Batching / batched design matrices	Processing large data in smaller chunks to reduce memory usage (e.g. In our GP stage, batching is used for kernel inversions; in our GAM stage, it limits design-matrix size during prediction).
Benchmarking	Systematic measurement of runtime, memory usage, and accuracy to quantify performance improvements between workflows.
Binary I/O and intermediates	Storing intermediate data in compact binary format rather than text, drastically reducing file size and read/write time.
Bounded per-task RAM	Explicitly limiting the memory available to each task so that many tasks can run concurrently without exceeding node capacity.
Concurrency	The number of tasks executed simultaneously on available computing resources.
Conda	An open source package and environment management system for Python that helps users install, update and manage software packages and dependencies.
Conda environment file (e.g. `environment.yml`)	Specification file listing all Python dependencies and versions for reproducible installation through Conda.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Container / containerisation	A portable computing environment (e.g. Docker, Singularity) that packages software (across programming languages) and dependencies to ensure identical execution across systems.
Container recipe	Configuration file (e.g. Dockerfile or Singularity definition) that builds the container environment to match the HPC setup.
CPU core	The smallest independent processor unit that executes individual tasks.
Cross-validation	A GAM evaluation technique used to select optimal smoothing by estimating how well the GAM generalizes to unseen data.
Deterministic file naming	A systematic naming convention (e.g. `latXXpXXX_lonXXpXXX`) that encodes coordinates and aids automatic tracking and recovery of task outputs.
Deterministic re-run	Re-execution of the workflow that reproduces identical outputs when given the same inputs and configuration.
Discrete smoothing	An optimisation in `mgcv::bam()` where continuous covariate values are discretised into bins before fitting. This reduces computation and memory demand with minimal loss of accuracy.
Error propagation	Situation where a single task failure could halt or invalidate entire job batches. Task isolation prevents error propagation.
Extended ensemble	A very large set of model outputs generated by sampling from a GP emulator (e.g. one million parameter combinations) beyond the original (221) PPE members.
fREML (Fast Restricted Maximum Likelihood)	An efficient algorithm for estimating smoothing parameters in GAMs that avoids repeated cross-validation and scales well to large data sets.
Filesystem churn / metadata traffic	Performance loss caused by creation of many small files on a shared filesystem; Can be reduced via pipeline optimisation.
Filesystem contention	Performance degradation that occurs when too many processes simultaneously read from or write to the same storage system.
GAM (Generalised Additive Model)	A regression framework that represents linear or nonlinear relationships between parameter values and/or variables using smooth functions.
Gigadata / high-dimensional output	Extremely large datasets with millions of observations or many predictor variables, typical of high-resolution Earth system models and data, or model ensembles.
GP (Gaussian Process)	A non-parametric Bayesian statistical model used to emulate output from complex numerical simulations.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

GP emulator	The trained Gaussian Process model used as a statistical surrogate for expensive model runs. Once trained (e.g. on PPE output), GP emulators can rapidly predict millions of output values (mean and variance) at unseen parameter combinations.
Grid box / grid point	A spatial unit of climate model domain defined by latitude and longitude coordinates.
HPC (High-Performance Computing)	Large computing systems consisting of many interconnected processors and nodes that allow parallel execution of computationally intensive workloads.
I/O (Input–Output)	Reading from or writing data to disk or memory. Efficient I/O handling is essential for performance at scale.
JASMIN / ARCHER	UK national HPC infrastructures used in this study; JASMIN hosts the LOTUS cluster at the Centre for Environmental Data Analysis (CEDA), and ARCHER is the UK's national supercomputing service.
Job array index	The unique identifier for a single task within a Slurm array, allowing direct mapping to a specific grid box and month.
Loop-based wrapper / serial loop	A sequential control structure in which each task is executed one after another within a shell or Python loop.
LOTUS cluster	HPC cluster within JASMIN used for all benchmarking experiments; provides thousands of CPU cores and high-speed interconnect.
Manifest script (`install_R_deps.R`)	The R script that installs required R packages and records `sessionInfo()`, and documents exact package versions used in the analysis.
Marginal importance/contribution	The result of variance decomposition analysis, where the marginal importance of each parameter x_i is the contribution to the modelled response variance after accounting for the influence of all other parameters in the full additive model. Represents the sensitivity of y to x_i within the multivariate GAM framework, not the effect of perturbing x_i alone.
Matrix operations / linear algebra kernels	Core computational routines (e.g. matrix multiplications, inversions) that dominate the cost of GP and GAM calculations. Parallelising or batching them greatly improves performance.
Median-hold method	The baseline approach for estimating parameter importance in GAMs. Each parameter x_i is varied across the ensemble while all others are held fixed at their median values. The variance of the resulting one-dimensional prediction series quantifies the isolated influence of x_i. Conceptually simple but computationally expensive, as it requires P (number of parameters in the PPE) independent prediction passes.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Memory footprint / peak memory	The amount of RAM used by a process; *peak memory* is the maximum observed usage during execution.
Memory reduction factor	Ratio of baseline to optimised peak memory usage.
Multithreading	Running multiple threads (lightweight tasks) simultaneously within a single process on a compute node, allowing different parts of a program to execute concurrently on separate cores.
Node	A physical compute unit within an HPC cluster that contains CPUs (and sometimes GPUs) and memory. Multiple tasks may run concurrently on one node.
Node-level contention	Competition among tasks running on the same HPC node for shared resources such as memory bandwidth, cache, or I/O channels. This contention can degrade performance and lead to superlinear scaling behaviour.
Numerical fidelity	The degree to which outputs from an optimised workflow exactly match those from a baseline implementation (e.g. Pearson $r = 1.000$).
OpenMP (v4.5)	A shared-memory parallel programming interface that allows multiple threads to run parts of a program simultaneously within one node.
Optimised workflow / pipeline	A redesigned workflow introducing task-level parallelism, efficient I/O, bounded memory, and reproducible automation to achieve large performance gains.
Parameter sensitivity	The strength or influence of each input parameter on model output (quantified here using GAM variance decomposition).
Parameter space	The multidimensional range of all uncertain model parameter combinations systematically explored by the PPE and emulated by the GP.
Partial smooth term (f_i)	A nonlinear smooth function in a GAM describing how model output changes with parameter x_i. Each smooth $f_i(x_i)$ is fitted jointly with the others and contributes a term $t_i = f_i(x_i)$ to the overall additive predictor $\eta = \sum_i f_i(x_i)$.
Pearson correlation	Statistical measure of linear correlation between two sets of values; used here to verify identical outputs between baseline and optimised workflows.
Per-task log	An individual log file (e.g. `slurm_output_%A_%a.out`) generated by each task, recording program outputs and error messages.
Pipeline vs. workflow	*Workflow*: the scientific sequence of operations (GP emulation → GAM fitting → analysis). *Pipeline*: the software system that automates and manages workflow reproducibly on HPC resources.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

PPE (Perturbed-Parameter Ensemble)	A collection of model simulations designed to efficiently span high-dimensional parameter space.
Post-check script	A lightweight diagnostic script that scans task logs to identify failed or incomplete tasks and automatically resubmits only those tasks.
rBAM (R Big Additive Model)	Implementation of GAM fitting using the R package `mgcv::bam()`, designed specifically for large data sets. It uses fast restricted maximum likelihood (fREML) estimation, discrete smoothing, and multithreading for efficiency.
Robustness	The ability of the pipeline to complete large numbers of tasks without failure or manual intervention.
Scalable design	Architecture that allows the workflow to expand from small test cases to full global or multi-model analyses with minimal code change.
Scalability / scaling test	Measurement of how runtime and memory usage change as the number of tasks or data size increases. Good scalability means near-linear increases in wall time as workload grows.
Slurm	An open-source workload manager used on many current HPC systems to schedule and monitor jobs across compute nodes. Version 22.05 was used here.
Slurm job array	A mechanism in Slurm that submits and manages large numbers of related tasks as a single batch job. Each array index runs one task (one grid box $\times$ month).
Sparse design matrices	Memory-efficient representation of matrices that store only non-zero elements, reducing memory load.
Speed-up factor	The ratio of baseline to optimised runtimes (e.g. 25$\times$ faster). Indicates performance improvement.
Superlinear scaling	A performance regime in which total runtime or memory increases faster than linearly with the number of concurrent tasks, often caused by node-level contention, shared resource limits, or excessive inter-process communication.
Task	The smallest independent computational unit in a pipeline (e.g. analysis in one model grid box for one month of data).
Task-centred folder layout	Directory structure in which each grid box/month task has its own subfolder containing configuration, logs, and outputs, facilitating parallel execution and recovery.
Task-level parallelism	Running many small, independent tasks concurrently to maximise HPC utilisation.
Task table	A structured list or index that maps each task (month, latitude, longitude combination) to its associated Slurm array index for organised execution and recovery.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Task-table orchestration	Replacement for loop-based execution in which a structured table of tasks is distributed automatically to many CPUs or nodes.
Throttling	Limiting the number of concurrent tasks in a job array to avoid overloading the file system or exceeding resource limits.
Throttling limit	The maximum number of simultaneous tasks allowed in a job array to prevent filesystem contention (typically 200–500 tasks).
Throughput	The total amount of computational work completed per unit time; improved here through concurrency and task isolation.
Variance decomposition	Statistical breakdown of the total output variance into contributions from individual model parameters and interactions between them.
Vectorised terms-based computation	The optimised method for GAM variances decomposition using `predict(..., type="terms")` from `mgcv::bam()`. It evaluates all smooth terms simultaneously, obtaining per-parameter contributions $t_i = f_i(x_i)$ and computing $\mathrm{Var}(t_i)$ and $\mathrm{sign}[\mathrm{Cov}(x_i, t_i)]$ in one batched pass. Numerically equivalent to the baseline median-hold approach but $\geq 20\times$ faster.
Wall time	The total elapsed real time taken for a job or task to complete, including computation and waiting time on shared resources.
YAML / JSON configuration files	Human-readable text files used to specify metadata (e.g. month, grid location, and variable name); useful in ensuring a reproducible configuration across runs.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Appendix B: Detailed variance contributions at different grid points

Table B1. Panel (a): lat=34.375, lon=-10.3125. Correlation $r = 1.0$, $\Delta = 9.325873 \times 10^{-15}$.

#	Parameter	pyGAM_var	rBAM_var	pyGAM%	rBAM%
1	bl_nuc	2.154827e-05	2.143380e-05	0.0159	0.0158
2	ait_width	3.732166e-05	3.720376e-05	0.0275	0.0274
3	cloud_ph	7.627990e-03	7.627270e-03	5.6153	5.6149
4	carb_ff_diam	7.579696e-06	7.466011e-06	0.0056	0.0055
5	carb_bb_diam	1.946541e-05	1.933888e-05	0.0143	0.0142
6	carb_res_diam	1.450057e-04	1.449373e-04	0.1067	0.1067
7	prim_so4_diam	4.479055e-04	4.478032e-04	0.3297	0.3297
8	sea_spray	5.801138e-05	5.792725e-05	0.0427	0.0426
9	anth_so2_chi	2.336507e-04	2.335902e-04	0.1720	0.1720
10	anth_so2_asi	8.859470e-05	8.847588e-05	0.0652	0.0651
11	anth_so2_eur	7.433930e-04	7.432857e-04	0.5472	0.5472
12	anth_so2_nam	2.217632e-07	1.846962e-07	0.0002	0.0001
13	anth_so2_r	3.849691e-03	3.849607e-03	2.8339	2.8340
14	volc_so2	8.508213e-03	8.508757e-03	6.2633	6.2639
15	bvoc_soa	5.358237e-05	5.350588e-05	0.0394	0.0394
16	dms	3.842745e-03	3.842300e-03	2.8288	2.8286
17	prim_moc	7.147707e-06	7.074799e-06	0.0053	0.0052
18	dry_dep_ait	1.015570e-05	1.002235e-05	0.0075	0.0074
19	dry_dep_acc	9.934354e-02	9.934227e-02	73.1312	73.1325
20	dry_dep_so2	6.798818e-03	6.798957e-03	5.0049	5.0052
21	kappa_oc	1.299390e-05	1.293028e-05	0.0096	0.0095
22	sig_w	1.743107e-05	1.737067e-05	0.0128	0.0128
23	rain_frac	3.232350e-05	3.221549e-05	0.0238	0.0237
24	cloud_ice_thresh	8.200467e-06	8.135611e-06	0.0060	0.0060
25	conv_plume_scav	1.897827e-04	1.898093e-04	0.1397	0.1397
26	bc_ri	1.236902e-04	1.237091e-04	0.0911	0.0911
27	oxidants_oh	2.036597e-03	2.036923e-03	1.4992	1.4995
28	oxidants_o3	1.595521e-04	1.594703e-04	0.1175	0.1174
29	bparam	1.963180e-05	1.956252e-05	0.0145	0.0144
30	two_d_fsd_factor	1.358385e-06	1.223127e-06	0.0010	0.0009
31	c_r_correl	1.305700e-06	1.202045e-06	0.0010	0.0009
32	autoconv_exp_lwp	2.142986e-04	2.142594e-04	0.1578	0.1577
33	autoconv_exp_nd	3.160195e-07	2.385999e-07	0.0002	0.0002
34	dbsdtbs_turb_0	9.449117e-05	9.433347e-05	0.0696	0.0694
35	ai	2.492509e-04	2.492142e-04	0.1835	0.1835
36	m_ci	7.353853e-04	7.351438e-04	0.5413	0.5412
37	a_ent_1_rp	1.016686e-04	1.016566e-04	0.0748	0.0748

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Table B2. Panel (b): lat=35.625, lon=6.5625. Correlation $r = 1.0$, $\Delta = 9.414691 \times 10^{-14}$.

#	Parameter	pyGAM_var	rBAM_var	pyGAM%	rBAM%
1	bl_nuc	4.842717e-07	1.138994e-07	0.0001	0.0000
2	ait_width	3.942348e-05	3.899612e-05	0.0086	0.0085
3	cloud_ph	1.040701e-01	1.040595e-01	22.8159	22.8150
4	carb_ff_diam	6.438358e-05	6.395226e-05	0.0141	0.0140
5	carb_bb_diam	1.985580e-04	1.979110e-04	0.0435	0.0434
6	carb_res_diam	5.207525e-04	5.202101e-04	0.1142	0.1141
7	prim_so4_diam	2.418347e-03	2.417737e-03	0.5302	0.5301
8	sea_spray	1.268690e-04	1.264933e-04	0.0278	0.0277
9	anth_so2_chi	6.793535e-07	3.794413e-07	0.0001	0.0001
10	anth_so2_asi	1.079060e-04	1.074695e-04	0.0237	0.0236
11	anth_so2_eur	3.186827e-03	3.186730e-03	0.6987	0.6987
12	anth_so2_nam	6.334871e-05	6.309090e-05	0.0139	0.0138
13	anth_so2_r	5.821229e-03	5.821005e-03	1.2762	1.2763
14	volc_so2	4.296915e-02	4.296921e-02	9.4204	9.4210
15	bvoc_soa	4.770095e-04	4.764368e-04	0.1046	0.1045
16	dms	4.135431e-03	4.133677e-03	0.9066	0.9063
17	prim_moc	1.067618e-04	1.063815e-04	0.0234	0.0233
18	dry_dep_ait	2.007294e-04	2.003857e-04	0.0440	0.0439
19	dry_dep_acc	2.755711e-01	2.755669e-01	60.4152	60.4178
20	dry_dep_so2	8.536680e-03	8.537633e-03	1.8715	1.8719
21	kappa_oc	3.449405e-04	3.447041e-04	0.0756	0.0756
22	sig_w	1.718250e-04	1.714976e-04	0.0377	0.0376
23	rain_frac	8.002712e-06	7.540826e-06	0.0018	0.0017
24	cloud_ice_thresh	2.180739e-04	2.177554e-04	0.0478	0.0477
25	conv_plume_scav	3.645826e-04	3.644498e-04	0.0799	0.0799
26	bc_ri	1.041745e-03	1.041701e-03	0.2284	0.2284
27	oxidants_oh	1.713233e-03	1.713951e-03	0.3756	0.3758
28	oxidants_o3	4.691158e-04	4.689323e-04	0.1028	0.1028
29	bparam	1.008724e-05	9.665671e-06	0.0022	0.0021
30	two_d_fsd_factor	3.010490e-04	3.004925e-04	0.0660	0.0659
31	c_r_correl	2.403138e-04	2.399419e-04	0.0527	0.0526
32	autoconv_exp_lwp	2.501608e-04	2.496439e-04	0.0548	0.0547
33	autoconv_exp_nd	6.119116e-06	5.833116e-06	0.0013	0.0013
34	dbsdtbs_turb_0	2.112376e-04	2.104993e-04	0.0463	0.0462
35	ai	2.508627e-04	2.505909e-04	0.0550	0.0549
36	m_ci	1.745613e-03	1.744752e-03	0.3827	0.3825
37	a_ent_1_rp	1.662003e-04	1.658816e-04	0.0364	0.0364

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Table B3. Panel (c): lat=36.875, lon=10.3125. Correlation $r = 1.0$, $\Delta = 1.221245 \times 10^{-15}$.

#	Parameter	pyGAM_var	rBAM_var	pyGAM%	rBAM%
1	bl_nuc	6.210545e-06	5.771778e-06	0.0012	0.0012
2	ait_width	3.197406e-05	3.143467e-05	0.0064	0.0063
3	cloud_ph	1.145467e-01	1.145435e-01	22.9790	22.9796
4	carb_ff_diam	6.259850e-04	6.253890e-04	0.1256	0.1255
5	carb_bb_diam	7.774527e-06	7.044340e-06	0.0016	0.0014
6	carb_res_diam	5.782129e-04	5.771827e-04	0.1160	0.1158
7	prim_so4_diam	6.603589e-05	6.553935e-05	0.0132	0.0131
8	sea_spray	7.689797e-04	7.686216e-04	0.1543	0.1542
9	anth_so2_chi	6.379148e-05	6.324752e-05	0.0128	0.0127
10	anth_so2_asi	1.411526e-04	1.405264e-04	0.0283	0.0282
11	anth_so2_eur	3.128465e-03	3.127743e-03	0.6276	0.6275
12	anth_so2_nam	3.673354e-05	3.634072e-05	0.0074	0.0073
13	anth_so2_r	3.342288e-03	3.342437e-03	0.6705	0.6706
14	volc_so2	5.368997e-02	5.369247e-02	10.7706	10.7717
15	bvoc_soa	6.209827e-04	6.200822e-04	0.1246	0.1244
16	dms	4.488922e-03	4.487707e-03	0.9005	0.9003
17	prim_moc	2.389569e-04	2.384999e-04	0.0479	0.0478
18	dry_dep_ait	1.664658e-06	7.949779e-07	0.0003	0.0002
19	dry_dep_acc	2.833205e-01	2.833109e-01	56.8363	56.8376
20	dry_dep_so2	1.023593e-02	1.023701e-02	2.0534	2.0537
21	kappa_oc	3.967965e-04	3.964665e-04	0.0796	0.0795
22	sig_w	4.418954e-04	4.414493e-04	0.0886	0.0886
23	rain_frac	9.311263e-05	9.245353e-05	0.0187	0.0185
24	cloud_ice_thresh	2.192088e-04	2.189360e-04	0.0440	0.0439
25	conv_plume_scav	4.464636e-04	4.462359e-04	0.0896	0.0895
26	bc_ri	1.216697e-03	1.216423e-03	0.2441	0.2440
27	oxidants_oh	4.582443e-03	4.583861e-03	0.9193	0.9196
28	oxidants_o3	2.279235e-03	2.278651e-03	0.4572	0.4571
29	bparam	2.041844e-05	2.003381e-05	0.0041	0.0040
30	two_d_fsd_factor	9.669467e-07	1.213981e-07	0.0002	0.0000
31	c_r_correl	9.337200e-04	9.328608e-04	0.1873	0.1871
32	autoconv_exp_lwp	4.418095e-04	4.406640e-04	0.0886	0.0884
33	autoconv_exp_nd	1.880826e-04	1.876115e-04	0.0377	0.0376
34	dbsdtbs_turb_0	6.984405e-04	6.966590e-04	0.1401	0.1398
35	ai	8.380924e-03	8.378274e-03	1.6813	1.6808
36	m_ci	2.178920e-03	2.179645e-03	0.4371	0.4373
37	a_ent_1_rp	2.474975e-05	2.438703e-05	0.0050	0.0049

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Table B4. Panel (d): lat=38.125, lon=2.8125. Correlation $r = 1.0$, $\Delta = 3.020917 \times 10^{-13}$.

#	Parameter	pyGAM_var	rBAM_var	pyGAM%	rBAM%
1	bl_nuc	8.062408e-07	4.110883e-07	0.0002	0.0001
2	ait_width	2.919628e-05	2.873685e-05	0.0076	0.0075
3	cloud_ph	9.703603e-02	9.702933e-02	25.3698	25.3700
4	carb_ff_diam	1.438795e-04	1.433741e-04	0.0376	0.0375
5	carb_bb_diam	5.720095e-06	5.179884e-06	0.0015	0.0014
6	carb_res_diam	6.884317e-04	6.877401e-04	0.1800	0.1798
7	prim_so4_diam	8.149727e-05	8.103385e-05	0.0213	0.0212
8	sea_spray	2.222658e-04	2.219318e-04	0.0581	0.0580
9	anth_so2_chi	2.806394e-05	2.776002e-05	0.0073	0.0073
10	anth_so2_asi	1.456823e-04	1.451813e-04	0.0381	0.0380
11	anth_so2_eur	2.924841e-03	2.924511e-03	0.7647	0.7647
12	anth_so2_nam	2.500791e-04	2.497382e-04	0.0654	0.0653
13	anth_so2_r	1.534950e-03	1.534738e-03	0.4013	0.4013
14	volc_so2	4.211497e-02	4.211555e-02	11.0109	11.0118
15	bvoc_soa	4.712641e-04	4.705617e-04	0.1232	0.1230
16	dms	3.922004e-03	3.920777e-03	1.0254	1.0252
17	prim_moc	8.333368e-05	8.293054e-05	0.0218	0.0217
18	dry_dep_ait	6.153514e-05	6.084035e-05	0.0161	0.0159
19	dry_dep_acc	2.082717e-01	2.082650e-01	54.4522	54.4545
20	dry_dep_so2	1.106182e-02	1.106306e-02	2.8921	2.8926
21	kappa_oc	3.261079e-04	3.257580e-04	0.0853	0.0852
22	sig_w	1.285097e-04	1.280960e-04	0.0336	0.0335
23	rain_frac	1.057164e-05	9.870116e-06	0.0028	0.0026
24	cloud_ice_thresh	3.204194e-06	2.911581e-06	0.0008	0.0008
25	conv_plume_scav	2.876564e-04	2.875143e-04	0.0752	0.0752
26	bc_ri	5.952172e-04	5.948458e-04	0.1556	0.1555
27	oxidants_oh	2.977803e-03	2.978755e-03	0.7785	0.7788
28	oxidants_o3	7.191476e-04	7.188698e-04	0.1880	0.1880
29	bparam	2.505360e-05	2.458776e-05	0.0066	0.0064
30	two_d_fsd_factor	2.605648e-05	2.538907e-05	0.0068	0.0066
31	c_r_correl	6.536703e-04	6.531370e-04	0.1709	0.1708
32	autoconv_exp_lwp	9.745337e-04	9.737073e-04	0.2548	0.2546
33	autoconv_exp_nd	3.687479e-05	3.647051e-05	0.0096	0.0095
34	dbsdtbs_turb_0	2.379767e-03	2.377619e-03	0.6222	0.6217
35	ai	3.163168e-03	3.162299e-03	0.8270	0.8268
36	m_ci	8.108417e-04	8.100102e-04	0.2120	0.2118
37	a_ent_1_rp	2.893822e-04	2.889307e-04	0.0757	0.0755

Consistency in parameter effect signs is confirmed in Fig. B1, where both workflows classify each parameter identically as positive, negative, or neutral. Therefore, the results show complete equivalence in scientific interpretation from both the well established baseline and the new optimised GAM approach.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

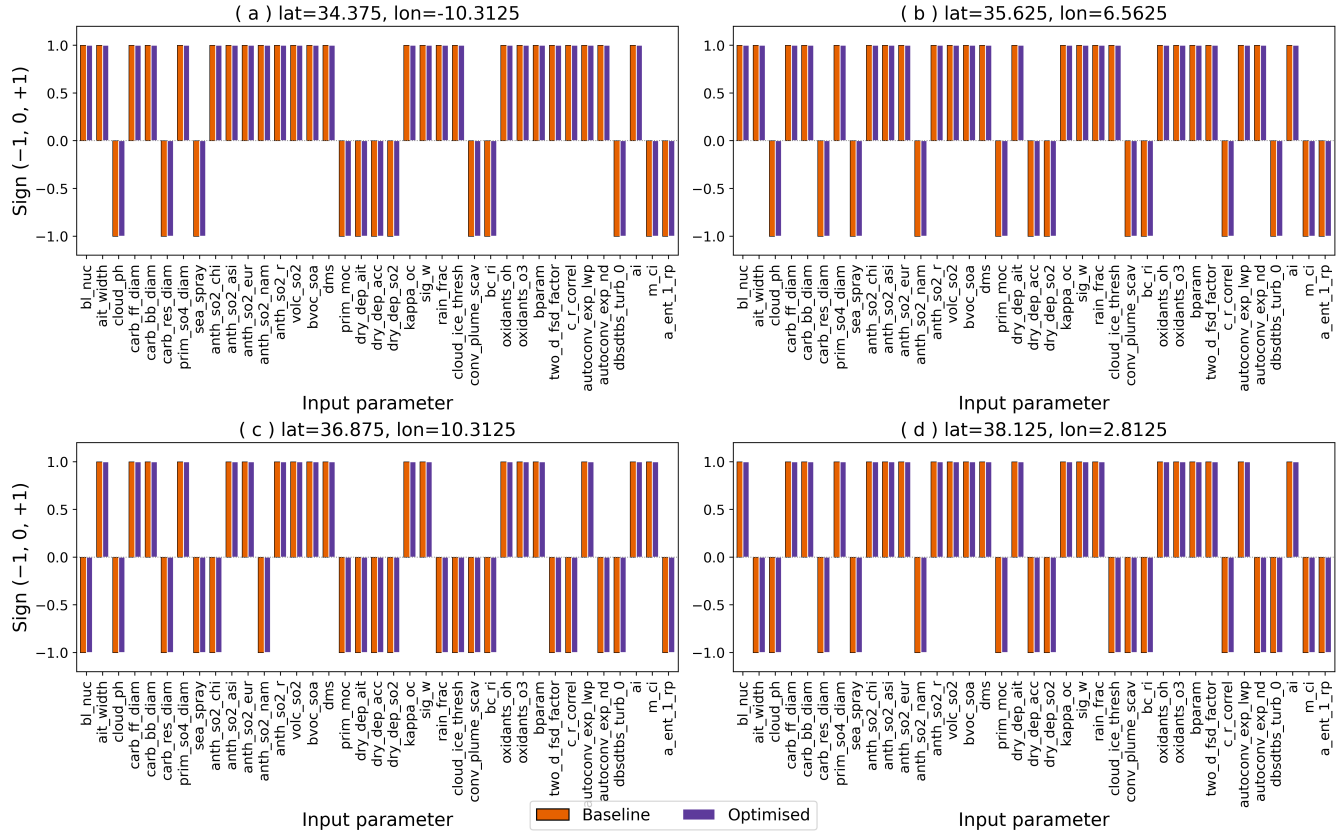

Figure B1. Parameter effect signs from GAM fitting for four different grid points. Orange: baseline (`pyGAM`); purple: optimised (rBAM).

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Appendix C: Runtime and peak memory usage

Table C1. Runtime and peak memory usage for baseline and optimised pipeline, by stage and end-to-end. Speed-up and memory-reduction factors are calculated as baseline/optimised.

Metric	GAM fitting	GP emulation	End-to-end
Baseline time (s)	10,623	6,177	16,800
Optimised time (s)	511	154	665
Speed-up (×)	20.8	40.1	25.3
Baseline peak mem. (GB)	∼100	∼50	∼100
Optimised peak mem. (GB)	∼10	∼6.5	∼10
Mem. reduction (×)	10.0	7.7	10.0

Notes: Times are median walltime across tasks; memory values are approximate peaks per task.

Author contributions. KG designed and implemented the optimised pipeline; executed the experiments; performed the analysis; and prepared all figures and tables. LAR developed the baseline workflow scripts and provided methodological input. KG and LAR jointly interpreted the results and wrote the manuscript; both authors reviewed and approved the final version.

Competing interests. The authors declare no competing interests.

Acknowledgements. We thank Prof. Ken Carslaw for invaluable discussions and guidance throughout the development of this work. We are also grateful to Jill Johnson, Jonathan Owen, Lea Prevost, Iain Webb, and Jeremy Oakley, for detailed feedback on early versions of the pipeline and manuscript. Additionally, we appreciate legacy code shared by Lindsay Lee and Jill Johnson which was entrained into the baseline workflow.

We acknowledge the use of the JASMIN super-data-cluster, managed by the UK Centre for Environmental Data Analysis (CEDA), which hosted the simulations and benchmarking experiments presented in this paper. The PPE that informed this research was created using the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk, last access 21 January 2021) under project allocation `n02-NEP013406`.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Financial support

We acknowledge funding from the UK Natural Environment Research Council (NERC) under grants **A-CURE** (NE/P013406/1) and **Aerosol-MFR** (NE/X013901/1). Earlier related collaborations that informed this work were supported by NERC grant NE/G006148/1 (AEROS) and the FORCES project under the European Union's Horizon 2020 research programme with grant agreement 821205. LR was supported by the Met Office Hadley Centre Climate Programme funded by DSIT.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

References

Bellouin, N., Quaas, J., Gryspeerdt, E., Kinne, S., Stier, P., Watson-Parris, D., Boucher, O., Carslaw, K. S., Christensen, M., Daniau, A.-L., Dufresne, J.-L., Feingold, G., Fiedler, S., Forster, P., Gettelman, A., Haywood, J. M., Lohmann, U., Malavelle, F., Mauritsen, T., McCoy, D. T., Myhre, G., Mülmenstädt, J., Neubauer, D., Possner, A., Rugenstein, M., Sato, Y., Schulz, M., Schwartz, S. E., Sourdeval, O., Storelvmo, T., Toll, V., Winker, D., and Stevens, B.: Bounding Global Aerosol Radiative Forcing of Climate Change, Reviews of Geophysics, 58, e2019RG000 660, https://doi.org/https://doi.org/10.1029/2019RG000660, e2019RG000660 10.1029/2019RG000660, 2020.

Carnell, R.: lhs: Latin Hypercube Samples, https://CRAN.R-project.org/package=lhs, r package version 1.1.6, 2022.

Carslaw, K. S., Regayre, L. A., Proske, U., Gettelman, A., Sexton, D. M. H., Qian, Y., Marshall, L., Wild, O., van Lier-Walqui, M., Oertel, A., Peatier, S., Yang, B., Johnson, J. S., Li, S., McCoy, D. T., Sanderson, B. M., Williamson, C. J., Elsaesser, G. S., Yamazaki, K., and Booth, B. B. B.: Opinion: The importance and future development of perturbed parameter ensembles in climate and atmospheric science, EGUsphere, 2025, 1–31, https://doi.org/10.5194/egusphere-2025-4341, 2025.

Durack, P. J., Taylor, K. E., Gleckler, P. J., Meehl, G. A., Lawrence, B. N., Covey, C., Stouffer, R. J., Levavasseur, G., Ben-Nasser, A., Denvil, S., Stockhause, M., Gregory, J. M., Juckes, M., Ames, S. K., Antonio, F., Bader, D. C., Dunne, J. P., Ellis, D., Eyring, V., Fiore, S. L., Joussaume, S., Kershaw, P., Lamarque, J.-F., Lautenschlager, M., Lee, J., Mauzey, C. F., Mizielinski, M., Nassisi, P., Nuzzo, A., O'Rourke, E., Painter, J., Potter, G. L., Rodriguez, S., and Williams, D. N.: The Coupled Model Intercomparison Project (CMIP): Reviewing project history, evolution, infrastructure and implementation, EGUsphere, 2025, 1–74, https://doi.org/10.5194/egusphere-2024-3729, 2025.

Johnson, J., Cui, Z., Lee, L., Gosling, J., Blyth, A., and Carslaw, K.: Evaluating uncertainty in convective cloud microphysics using statistical emulation, Journal of Advances in Modeling Earth Systems, 7, 162–187, 2015.

Johnson, J. S., Regayre, L. A., Yoshioka, M., Pringle, K. J., Lee, L. A., Sexton, D. M. H., Rostron, J. W., Booth, B. B. B., and Carslaw, K. S.: The importance of comprehensive parameter sampling and multiple observations for robust constraint of aerosol radiative forcing, Atmospheric Chemistry and Physics, 18, 13 031–13 053, https://doi.org/10.5194/acp-18-13031-2018, 2018.

Johnson, J. S., Regayre, L. A., Yoshioka, M., Pringle, K. J., Turnock, S. T., Browse, J., Sexton, D. M. H., Rostron, J. W., Schutgens, N. A. J., Partridge, D. G., Liu, D., Allan, J. D., Coe, H., Ding, A., Cohen, D. D., Atanacio, A., Vakkari, V., Asmi, E., and Carslaw, K. S.: Robust observational constraint of uncertain aerosol processes and emissions in a climate model and the effect on aerosol radiative forcing, Atmospheric Chemistry and Physics, 20, 9491–9524, https://doi.org/10.5194/acp-20-9491-2020, 2020.

Knutti, R.: Should we believe model predictions of future climate change?, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366, 4647–4664, 2008.

Lawrence, B. N., Bennett, V. L., Churchill, J., Juckes, M., Kershaw, P., Pascoe, S., Pepler, S., Pritchard, M., and Stephens, A.: Storing and manipulating environmental big data with JASMIN, in: 2013 IEEE international conference on big data, pp. 68–75, IEEE, 2013.

Lee, L. A., Carslaw, K. S., Pringle, K. J., Mann, G. W., and Spracklen, D. V.: Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters, Atmospheric Chemistry and Physics, 11, 12 253–12 273, https://doi.org/10.5194/acp-11-12253-2011, 2011.

Lee, L. A., Carslaw, K. S., Pringle, K. J., and Mann, G. W.: Mapping the uncertainty in global CCN using emulation, Atmospheric Chemistry and Physics, 12, 9739–9751, https://doi.org/10.5194/acp-12-9739-2012, 2012.

Lee, L. A., Reddington, C. L., and Carslaw, K. S.: On the relationship between aerosol model uncertainty and radiative forcing uncertainty, Proceedings of the National Academy of Sciences, 113, 5820–5827, https://doi.org/10.1073/pnas.1507050113, 2016.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Mersmann, O.: truncnorm: Truncated Normal Distribution, https://CRAN.R-project.org/package=truncnorm, r package version 1.0-9, 2022.

Oakley, J. and O'hagan, A.: Bayesian inference for the uncertainty distribution of computer model outputs, Biometrika, 89, 769–784, 2002.

O'Hagan, A.: Bayesian analysis of computer code outputs: A tutorial, Reliability Engineering System Safety, 91, 1290–1300, https://doi.org/https://doi.org/10.1016/j.ress.2005.11.025, the Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004), 2006.

Prévost, L. M. C., Regayre, L. A., Johnson, J. S., McNeall, D., Milton, S., and Carslaw, K. S.: Detection of structural deficiencies in a global aerosol model to explain limits in parametric uncertainty reduction, EGUsphere, 2025, 1–65, https://doi.org/10.5194/egusphere-2025-4795, 2025.

Pujol, G., Iooss, B., and Janon, A.: sensitivity: Global Sensitivity Analysis of Model Outputs, https://CRAN.R-project.org/package=sensitivity, r package version 1.18.1, 2017.

R Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/, 2022.

Regayre, L. A., Pringle, K. J., Booth, B. B. B., Lee, L. A., Mann, G. W., Browse, J., Woodhouse, M. T., Rap, A., Reddington, C. L., and Carslaw, K. S.: Uncertainty in the magnitude of aerosol-cloud radiative forcing over recent decades, Geophysical Research Letters, 41, 9040–9049, https://doi.org/https://doi.org/10.1002/2014GL062029, 2014.

Regayre, L. A., Deaconu, L., Grosvenor, D. P., Sexton, D. M. H., Symonds, C., Langton, T., Watson-Paris, D., Mulcahy, J. P., Pringle, K. J., Richardson, M., Johnson, J. S., Rostron, J. W., Gordon, H., Lister, G., Stier, P., and Carslaw, K. S.: Identifying climate model structural inconsistencies allows for tight constraint of aerosol radiative forcing, Atmospheric Chemistry and Physics, 23, 8749–8768, https://doi.org/10.5194/acp-23-8749-2023, 2023.

Regayre, L. A., Prévost, L. M. C., Ghosh, K., Johnson, J. S., Oakley, J. E., Owen, J., Webb, I., and Carslaw, K. S.: Remaining aerosol forcing uncertainty after observational constraint and the processes that cause it, EGUsphere, 2025, 1–50, https://doi.org/10.5194/egusphere-2025-3755, 2025.

Roustant, O., Ginsbourger, D., and Deville, Y.: DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization, Journal of Statistical Software, 51, 1–55, https://doi.org/10.18637/jss.v051.i01, 2012.

Sellar, A. A., Jones, C. G., Mulcahy, J. P., Tang, Y., Yool, A., Wiltshire, A., O'Connor, F. M., Stringer, M., Hill, R., Palmieri, J., Woodward, S., de Mora, L., Kuhlbrodt, T., Rumbold, S. T., Kelley, D. I., Ellis, R., Johnson, C. E., Walton, J., Abraham, N. L., Andrews, M. B., Andrews, T., Archibald, A. T., Berthou, S., Burke, E., Blockley, E., Carslaw, K., Dalvi, M., Edwards, J., Folberth, G. A., Gedney, N., Griffiths, P. T., Harper, A. B., Hendry, M. A., Hewitt, A. J., Johnson, B., Jones, A., Jones, C. D., Keeble, J., Liddicoat, S., Morgenstern, O., Parker, R. J., Predoi, V., Robertson, E., Siahaan, A., Smith, R. S., Swaminathan, R., Woodhouse, M. T., Zeng, G., and Zerroukat, M.: UKESM1: Description and Evaluation of the U.K. Earth System Model, Journal of Advances in Modeling Earth Systems, 11, 4513–4558, https://doi.org/https://doi.org/10.1029/2019MS001739, 2019.

Servera, J. V., Martino, L., Verrelst, J., and Camps-Valls, G.: Multifidelity Gaussian process emulation for atmospheric radiative transfer models, IEEE Transactions on Geoscience and Remote Sensing, 61, 1–10, 2023.

Servén, D., Brummitt, C., Abedi, H., and Hlink: dswah/pyGAM: v0.8.0, https://doi.org/10.5281/ZENODO.1208723, accessed: 2025-07-01, 2018.

Sottile, G. and Gohel, D.: trapezoid: Trapezoidal Distribution, https://CRAN.R-project.org/package=trapezoid, r package version 0.6, 2022.

Stouffer, R. J., Eyring, V., Meehl, G. A., Bony, S., Senior, C., Stevens, B., and Taylor, K.: CMIP5 scientific gaps and recommendations for CMIP6, Bulletin of the American Meteorological Society, 98, 95–105, 2017.

https://doi.org/10.5194/egusphere-2025-5533
Preprint. Discussion started: 21 December 2025

Susiluoto, J., Spantini, A., Haario, H., Härkönen, T., and Marzouk, Y.: Efficient multi-scale Gaussian process regression for massive remote sensing data with satGP v0. 1.2, Geoscientific Model Development, 13, 3439–3463, 2020.

Watson-Parris, D., Williams, A., Deaconu, L., and Stier, P.: Model calibration using ESEm v1.1.0 – an open, scalable Earth system emulator, Geoscientific Model Development, 14, 7659–7672, https://doi.org/10.5194/gmd-14-7659-2021, 2021.

Wickham, H. and Bryan, J.: readr: Read Rectangular Text Data, https://CRAN.R-project.org/package=readr, r package version 2.1.4, 2023.

Wood, S. N.: Inference and computation with generalized additive models and their extensions, Test, 29, 307–339, 2020.

Wood, S. N., Goude, Y., and Shaw, S.: Generalized additive models for large data sets, Journal of the Royal Statistical Society Series C: Applied Statistics, 64, 139–155, 2015.

Yang, Q., Elsaesser, G. S., van Lier-Walqui, M., and Eidhammer, T.: A simple emulator that enables interpretation of parameter-output relationships, applied to two climate model PPEs, Journal of Advances in Modeling Earth Systems, 17, e2024MS004 766, 2025.

Yoshioka, M., Regayre, L. A., Pringle, K. J., Johnson, J. S., Mann, G. W., Partridge, D. G., Sexton, D. M. H., Lister, G. M. S., Schutgens, N., Stier, P., Kipling, Z., Bellouin, N., Browse, J., Booth, B. B. B., Johnson, C. E., Johnson, B., Mollard, J. D. P., Lee, L., and Carslaw, K. S.: Ensembles of Global Climate Model Variants Designed for the Quantification and Constraint of Uncertainty in Aerosols and Their Radiative Forcing, Journal of Advances in Modeling Earth Systems, 11, 3728–3754, https://doi.org/https://doi.org/10.1029/2019MS001628, 2019.