Author response to the referee's final comments on the Ghosh and Regayre manuscript entitled "Optimizing Gaussian Process Emulation and Generalized Additive Model Fitting for Rapid, Reproducible Earth System Model Analysis." We thank the reviewer for the positive assessment and for the two helpful final suggestions, which have improved the clarity of the manuscript. Reviewer comments are reproduced in blue, followed by our responses in black. Revisions to the manuscript are indicated in red.

Reviewer 1

The authors have done a good job of addressing the previous referees' comments, and I have only two minor things that they might choose to address in their final submission:

We address the two points in turn below.

1. P6, L150. Figure 2 shows the run time, but not the memory. It'd be rewrite this to be clear about that (it is not necessary to have a figure for the memory, but the text should not imply that figure 2 does show that).

We thank the reviewer for pointing this out. We agree that the previous wording could imply that Fig. 2 presents both runtime and memory use. We have revised the sentence so that Fig. 2 is referenced only for runtime; peak memory use is described separately in the following text, where we note that the baseline pyGAM implementation regularly exceeded 80 GB per process during spline-matrix assembly.

Revised text (Sect. 2.2.2, Baseline GAM fitting): "Figure 2 summarises the runtime profile of the baseline workflow and shows that GAM fitting for one million combinations across 37 parameter dimensions dominated total runtime. GAM fitting was also the main contributor to peak memory use."

2. I am confused about the 200 core concurrency limit, what drives that, is it just the available compute/scheduler architecture? Is this somehow related to P13L290? There are several things here I don't understand. In slurm you can request the memory resources you need, there are other compute types on JASMIN/LOTUS I believe ... and why can't you run batches in parallel as well? (In which case you have much more memory available, but still note enough to have 16 GB per task for 200 tasks which is what they say they need). If this was a "production configuration choice" the fine, but please say so. I've missed something here, but maybe that's my problem and I just missed the key sentence.

We thank the reviewer for identifying this ambiguity, and confirm that 200 was a production configuration choice rather than a fundamental limit of Slurm, JASMIN, or the workflow. We address the specific points below.

What drives the value of 200. The value of 200 is a Slurm-array throttle that we set deliberately for our benchmarking runs, chosen after preliminary testing to provide a stable and reproducible operating point on the shared LOTUS system. It sets a maximum of 200 concurrently running array tasks. Approximately 200 tasks generally ran concurrently, although the number could occasionally be lower depending on resource availability and system load. It is not a hard limit imposed by the scheduler, hardware, or method.

Relation to P13, L290, and per-task memory. The 10–16 GB value refers to the maximum memory requested for each individual task, providing sufficient headroom for variation among grid cells and cases. It should not be interpreted as the amount of memory continuously consumed by every task. Actual memory use was substantially lower and varied between tasks. Consequently, approximately 200 tasks could generally run concurrently under the production configuration. We have revised the passage at P13, L290 to distinguish clearly between requested memory and actual memory use.

Other compute types and parallel batches. We agree that higher-memory partitions exist on JASMIN/LOTUS and that, in principle, more resources could be requested. Under the allocation used for these benchmarks, however, simultaneous access to additional resources was limited, so submitting further batches concurrently did not reliably add capacity. On a system or allocation providing greater and more predictable resources, the throttle could be increased, or independent job arrays could be executed concurrently, allowing effective concurrency beyond 200. We have clarified these points in the manuscript.

Revised text (Sect. 2.1, Benchmarking setup): "Task submission was controlled using the #SBATCH --array directive with a Slurm-array throttle (e.g. --array=0-N%200). The value of 200 used throughout our scaling benchmarks was a deliberate production configuration choice rather than a hard limit of Slurm, the hardware, or the workflow. This throttle permits a maximum of 200 array tasks to run concurrently. Approximately 200 tasks generally executed concurrently under the benchmark configuration, although the number could occasionally be lower depending on resource availability and system load. The 10–16 GB value denotes the maximum memory requested per task and provides headroom for variability among grid cells and cases; actual memory consumption was lower and varied between tasks."

Revised text (Sect. 3.1, Optimised Gaussian process emulation stage):

"Task counts above 200 were executed in successive groups under this production configuration. The 200-task concurrency throttle was selected following preliminary testing as a stable and reproducible operating point on the shared LOTUS system. Approximately 200 tasks generally ran concurrently, whereas higher settings occasionally increased resource and filesystem contention and caused individual tasks to fail during periods of high system load."

Revised text (Sect. 3.2, Optimised GAM fitting stage):

"The 10–16 GB value represents the maximum memory requested per task and provides operational headroom for variation among grid cells and cases; it does not represent the memory continuously consumed by every task. Actual memory use was lower and varied between tasks, while approximately 200 tasks generally ran concurrently under the production configuration. Variability in resource availability and filesystem load on the shared LOTUS system nevertheless contributed to fluctuations in walltime, speed-up, and parallel efficiency."