JOURNAL OF REPRODUCIBLE STATISTICS

Response to the Editor and Reviewers

First revision • Minor revision requested

Manuscript
libmatern: Fast and Accurate Matérn Covariance Evaluation on SIMD CPUs and CUDA GPUs

Dear Editor and Reviewers,

We thank the editor and the three reviewers for their careful reading and constructive comments. In the revised manuscript, the timing tables report medians of 25 repetitions with interquartile ranges, and the captions give speedup bounds obtained by pairing opposite quartiles of the two implementations. The CPU headline is given together with its range over datasets and batch sizes, and the build configuration, determinism, upstream versions and the block timing of the size ladders are documented. The non-finite results and the (h, ν, s) comparison are now part of the main results, the figures use a colour-blind-safe palette, Table B13 maps every table and figure to the command that regenerates it, and `scipy.special.kv` was evaluated as a further CPU comparator. Section 6 estimates the covariance-evaluation share of a likelihood evaluation. Our responses to the individual points follow; section, table and figure numbers refer to the revised manuscript.

Authors: Daniel Mann, Matthis Ehrhardt and Martin Schlather Date: November 6, 2026

1 Decision letter and editorial requests

Subject: JRepStat: revision requested (minor)

Thank you for your submission to JRepStat. We are inviting a minor revision. The reviewers' points follow, grouped by weight.

libmatern evaluates the normalised Matérn covariance directly by folding the Gamma-power prefactor into several numerical formulations of the modified Bessel function K_ν, with architecture-specific routing for scalar CPU, SIMD CPU and CUDA GPU. It matters to spatial-statistics and Gaussian-process practitioners who repeatedly assemble Matérn covariance matrices, especially when the smoothness parameter is estimated, because it returns finite, accurate values in regimes where common libraries overflow and delivers large CPU throughput gains. We would be glad to see the revised version.

How to read this letter. The decision rests on the items marked required. If a required item seems wrong to you, or cannot be done within this paper, decline it on the revision page and give the reason. A reasoned decline is a legitimate answer, and the editor weighs it. Reviewers can be mistaken. Items marked recommended or optional are advice from the reviewers. Adopt them, decline them with a sentence, or leave them for later work. None of that counts against the paper. The reviewer reports below carry the same marks. The editor who signs this decision may weigh an item differently in the accompanying message, and that message prevails.

Authors' response

We thank the editor for the careful handling of the manuscript and for the clear guidance on how to read the letter. We have addressed all required items. Of the recommended and optional items, we give reasons for keeping the finiteness result out of the abstract (ED-REC-06), for leaving further hardware to later work (ED-OPT-01, ED-OPT-04) and for replacing a complete estimation loop by an estimate (ED-OPT-03); all other items have been adopted in some form.

1.1 Required editorial items

ED-REQ-01 REQUIRED

> [methods-soundness] The timing tables (Table 3, Table 4, Table B10) report a single "ns/-value" for each method, and the speedup claims rest entirely on these numbers, but the central tendency (mean, median) and the spread across the 25 measured repetitions are not stated. Please name the statistic used and add a measure of spread from the runs you already have, or qualify the speedup figures as point estimates from a single summary. Figure A5 already shows IQR and whiskers for the transfer strategies, so the same can be given for the headline comparisons.

Authors' response

All reported costs are now stated to be medians of 25 measured repetitions after two warm-up calls, and Tables 3, 4 and B10 give interquartile ranges (IQR) of the timings in per cent of the median. Section 4.4 defines speedup bounds, which pair the lower quartile of one implementation with the upper quartile of the other; the captions state that they differ from the speedups by less than 0.2% on the CPU (Table 3) and by at most 0.12 (kernel) and 0.02 (end-to-end) on the GPU.

Changes in the manuscript / artifacts: Section 4.4; Tables 3, 4 and B10 (IQR columns and captions).

ED-REQ-02 REQUIRED

> Reviewer 1: The abstract and Section 5.3 headline the CPU gain as the maximum $6.6\times$, drawn from the random mixture that Table B5 states was built to cover routing regions, not application workloads. Report a typical CPU speedup alongside the maximum and name the benchmark nature, so an application reader does not expect $6.6\times$ in their own pipeline.

Authors' response

The abstract and Section 5.3 now present $6.6\times$ as the single-threaded speedup for a synthetic random mixture of 2^{25} evaluations and report, alongside it, the range over the eight benchmark datasets ($3.9\times$ to $21\times$, median $7.5\times$). Section 5.3 adds the range over batch sizes and the value for fixed $\nu = 2.5$, and states that the random mixture was designed to cover the routing regions (Table B5), so that speedups in applications depend on their parameter distribution and batch sizes.

Changes in the manuscript / artifacts: Abstract; Section 5.3.

ED-REQ-03 REQUIRED

> Reviewer 2: State which summary statistic over the 25 repetitions the per-element costs in Tables 3, 4 and B10 report, and give a dispersion measure for the headline speedups, so the numbers are interpretable.

Authors' response

See ED-REQ-01.

Changes in the manuscript / artifacts: Section 4.4; Tables 3, 4 and B10.

ED-REQ-04 REQUIRED

> Reviewer 3: record the exact build configuration the reported ULP and timing numbers depend on, and attest that a repeat run reproduces them.

Authors' response

Section 4.4 and Table B7 now give the compilers, optimisation levels, instruction-set flags and floating-point contraction settings for every backend and for the CUDA code; all results were computed with `libmatern` 0.9.5 at commit `e370401`. Each point is evaluated independently, so its result does not depend on batch composition, order or thread count; accordingly, the per-point outputs are byte-identical on both CPUs and across separate rebuilds within the CPU and GPU runs. A repeat of all nine runs at the same commit, on other nodes of the same types, reproduced every accuracy statistic exactly; the median costs of at least 98% of the timing rows on each machine agreed within ± 10%, with larger differences for very short GPU kernels, in host–device transfers and at a few individual points (Section 4.4).

Changes in the manuscript / artifacts: Section 4.4; Table B7.

ED-REQ-05 REQUIRED

> Reviewer 3: pin the exact upstream version or commit of CuBesselK, and of every baseline, that the modification script and the ports target, since the manuscript states the CuBesselK source is downloaded separately and not redistributed.

Authors' response

Table B12 lists the upstream version or commit of every baseline and the packages used in the reported runs. CuBesselK has no tagged release; the modification script downloads its two source files at commit `94ec9ca` and checks them against SHA-256 digests (Section 4.2). The bibliography entries for CuBesselK and the Bessels.jl main branch point to the pinned commits.

Changes in the manuscript / artifacts: Section 4.2; Table B12; bibliography.

ED-REQ-06 REQUIRED

> Reviewer 3: state what the headline "maximum $6.6\times$ speedup" is taken over, and give the representative CPU speedup beside it, or qualify the abstract to call it a best case.

Authors' response

The word "maximum" has been removed. The abstract and Section 5.3 now state that $6.6\times$ is the speedup of the AVX-512 implementation over the fastest external implementation at $N = 2^{25}$ on the random mixture. It is also the median over the 18 batch sizes of the ladder (range $2.2\times$ to $6.8\times$) and therefore not an exceptionally best case; representative values are given as described under ED-REQ-02.

Changes in the manuscript / artifacts: Abstract; Section 5.3.

1.2 Recommended editorial items

ED-REC-01 RECOMMENDED

> [statistical-rigor] The speedup factors are ratios of two measured timings, so their uncertainty is larger than that of either timing alone, and a ratio near $1.00\times$ (the grid end-to-end entry in Table 4) may be compatible with no difference once variability is included. State the central speedup with the range implied by the measured timing spread, so a reader can see which crossovers and which near-unity ratios are resolved by the data and which are not.

Authors' response

The captions of Tables 3, 4 and B10 now give the speedup bounds defined in Section 4.4, which pair the lower quartile of one implementation with the upper quartile of the other; only those of the grid end-to-end ratio (0.99 to 1.01) include parity. For both GPU–CPU crossovers, the interquartile ranges of the GPU and CPU costs at the ladder sizes on either side do not overlap (Section 5.4).

Changes in the manuscript / artifacts: Captions of Tables 3, 4 and B10; Section 4.4. and 5.4.

ED-REC-02 RECOMMENDED

> [visualisation] Figures A4, A10, A11, A14 and A15 separate nine or ten implementations, and the overview panels A10/A11/A16/A17 and the box plots A12/A13 do the same. Distinguishing that many series by hue alone is hard for readers with colour vision deficiency and in greyscale. Pair each series with a distinct marker or line style, or draw from a colourblind-safe qualitative palette (for example Okabe-Ito, which supports up to eight categories), and consider whether every baseline needs its own line in a single panel. Figure 2 already does this well with circles, triangles and squares, so the fix is to extend that practice.

Authors' response

All figures were revised. They use the Okabe–Ito palette, with the AVX-512 and CUDA implementations of `libmatern` in vermillion, and distinguish the implementations additionally by marker; in the line plots (Figures A4, A14 and A15), library backends are drawn with solid and external implementations with dashed lines, and the rows of the box plots (Figures A12 and A13) are labelled by name. To reduce the number of series per panel, Figures A4 and A15 omit the scalar backend and Boost (their costs are in Table 3), Figure A14 omits the scalar and matched+FMA backends, and Figures A10–A13 omit kaust-40, whose maximum error exceeds 10^6 ULP on every dataset; the captions state where the omitted series lie.

Changes in the manuscript / artifacts: All figures; captions of Figures A4 and A10–A15.

ED-REC-03 RECOMMENDED

> [visualisation] Figure 1 identifies five algorithm regions (Temme, continued fractions, large-argument, large-order, Levin) plus hatched fused-exponent assembly by fill colour. The in-panel text labels help, but verify the region fills are distinguishable for colourblind readers and in print, since the hatching for fused-exponent assembly is the only non-colour encoding. The same applies to Figures A1, A8 and A9.

Authors' response

In Figures 1 and A1, every region now carries an in-panel label, so the regions remain identifiable in greyscale. In addition to the hatched fused-exponent assembly, the Temme region (Figure 1) and the power-series region (Figure A1) are dotted, and the borders in Figure 1 are distinguished by line style. In Figures A8 and A9, the assemblies differ by fill and hatching (direct: solid, logarithmic: hatched, scaled: dotted), and the `libmatern` value is a dashed line.

Changes in the manuscript / artifacts: Figures 1, A1, A8 and A9.

ED-REC-04 RECOMMENDED

> [writing-quality] A few phrasings sit below the register of the surrounding prose: “the conversion from K_ν to M_ν remains unfortunate” (Section 4.2 and again near Figures A8/A9) and “As a tiny step in this direction” (Section 6). Restating these in neutral terms would match the rest of the paper.

Authors’ response

Both phrasings were revised: “remains unfortunate” now reads “remains a source of error”, and the paragraph beginning “As a tiny step” was moved to Section 5.2 and rewritten (see ED-REC-07).

Changes in the manuscript / artifacts: Sections 4.2, 5.2 and 6.

ED-REC-05 RECOMMENDED

> [software-landscape] scipy.special.kv (the Cephes-derived evaluator behind scikit-learn’s Matern kernel, GPy and GPyTorch) is the dominant Matern/Bessel implementation in the Python and machine-learning ecosystem and is absent from the CPU baselines. Adding it as one further CPU comparator on the existing eight datasets, or stating in Section 4.2 why the C/C++/Julia set already represents the fastest available evaluators, would close the one remaining gap in an otherwise complete landscape. The paper stands without it.

Authors’ response

A clarification: SciPy documents `scipy.special.kv` as a wrapper of Amos’s routine `zbesk` (ACM TOMS Algorithm 644), not as Cephes-derived. It is the evaluator behind scikit-learn’s `Matern` kernel for orders other than $1/2$, $3/2$ and $5/2$, whereas GPyTorch and GPy evaluate closed forms for these orders.

We have evaluated `scipy.special.kv` on the eight datasets. To compare the evaluator rather than the Python layer, we compiled the C++ code that `kv` and `kve` of SciPy 1.18.1 call, unchanged, into the benchmark harness; without floating-point contraction it reproduces SciPy’s values bit for bit. It was assembled, selected, scored and timed like the other external implementations, on the AMD host.

SciPy returns non-finite values on 12.4% of the small-z and 32.7% of the large-ν points in the best assembly, similar to the Bessels.jl ports and Boost. On the other six datasets its maximum error is 186 to 1458 ULP, against 10.4 to 22.4 ULP for `libmatern`. With 256 to 471 ns per value it ranks third to fifth among the nine external CPU implementations and is 1.9 to 5.1 times slower than the fastest of them on every dataset, so none of the reported speedups changes. Its only effect on the summary statistics would be a lower external maximum on the half-integer set (907 instead of GSL’s 1248 ULP). Section 4.2 now states these results.

Changes in the manuscript / artifacts: Section 4.2; Table B12; bibliography.

ED-REC-06 RECOMMENDED

> Reviewer 1: Foreground the non-finite result: the Bessels.jl and Boost ports return roughly 12% non-finite covariances on the small-z set and 32% on the large-ν set, while libmatern returns none (Table B6). This is the clearest consequence for a practitioner and currently sits only in an appendix table.

Authors' response

Section 5.2 now gives the shares of non-finite results (about 12% on the small-z set and 32% on the large-ν set) and notes that such values propagate into any covariance matrix assembled from them. We report this result in Section 5.2 rather than in the abstract and introduction because finite output is not specific to `libmatern` on these sets: GSL and the modified CuBesselK port also return finite values throughout.

Changes in the manuscript / artifacts: Section 5.2.

ED-REC-07 RECOMMENDED

> Reviewer 1: Promote the (h, ν, s) interface comparison from the Discussion, since practitioners call the function with (h, ν, s), and that is where libmatern's accuracy edge is cleanest and least confounded by argument rounding.

Authors' response

The (h, ν, s) comparison has been moved from Section 6 to Section 5.2, which now states that the (z, ν) results compare the numerical methods and the (h, ν, s) results the interface that applications call.

Changes in the manuscript / artifacts: Sections 5.2 and 6; captions of Figures A11, A13 and A17.

ED-REC-08 RECOMMENDED

> Reviewer 1: Make the Gaussian-process relevance concrete using Table B11, with a short estimate of the covariance-evaluation share of a likelihood loop where ν is estimated.

Authors' response

Section 6 now contains this estimate. In the non-stationary setting of Table B11, a dense Gaussian likelihood on $n = M^2$ sites requires $E \approx M^4/2$ covariance values per iteration, all of which change when ν is estimated, and a Cholesky factorisation of about $n^3/3$ operations. With the measured costs per value (Table 3) and assumed single-core factorisation rates of 30 to 70 GFLOP/s, covariance evaluation takes 93 to 97% of an iteration at $M = 23$ and 46 to 66% at $M = 91$ with the fastest external implementation, against 67 to 82% and 11 to 23% with `libmatern`; a likelihood iteration thus becomes about 4.7 to 5.6 and 1.6 to 2.3 times faster. In the stationary settings of Table B11, the evaluation share of a dense likelihood is negligible.

Changes in the manuscript / artifacts: Section 6.

ED-REC-09 RECOMMENDED

> Reviewer 2: Report the speedup distribution or median across the size ladder alongside the maximum, since the abstract leads with the most favourable single point.

Authors' response

Section 5.3 now reports the speedup over all 18 batch sizes of the ladder ($2.2\times$ at $N = 256$ to $6.8\times$ at $N = 2^{16}$) and its median ($6.6\times$), which equals the value at $N = 2^{25}$ quoted in the abstract.

Changes in the manuscript / artifacts: Section 5.3.

ED-REC-10 RECOMMENDED

> Reviewer 2: State the compiler optimisation flags and FMA-contraction settings used for the accuracy builds and the timing builds, which govern both the ULP figures and the runtimes.

Authors' response

Section 4.4 and Table B7 now state the optimisation level, instruction-set flags and floating-point contraction for each library backend (`-O2 -ffp-contract=fast`; `-mfma` for AVX2, AVX-512 and matched+FMA), for the baselines (`-march=native -O2`) and for the CUDA code (`-O2 -arch=sm_80`, default FMA contraction). Accuracy and timings of each implementation are obtained with the same executable on the same points.

Changes in the manuscript / artifacts: Section 4.4; Table B7.

ED-REC-11 RECOMMENDED

> Reviewer 2: Add a determinism attestation for the SIMD and GPU reduction orders and a single command that reproduces the benchmark tables, and point to the captured environment.

Authors' response

Section 4.4 now attests determinism: each point is evaluated by one SIMD lane or one CUDA thread without reductions across points, which the test suite checks bit for bit, and the per-point outputs are byte-identical on both CPUs and across rebuilds. The run directories are attached to release v0.9.6 (`libmatern-0.9.6-runs.tar.gz`); with the code of release v0.9.6, `make figures` and `make tables` regenerate the figures and tables of the revised manuscript from them, and `make reproduce` repeats the benchmarks. Each run directory records its environment in `PROVENANCE.txt` and `MANIFEST.txt`.

Changes in the manuscript / artifacts: Section 4.4; Software and data availability; Table B13.

ED-REC-12 RECOMMENDED

Reviewer 2: Clarify in Table 3 how the N=2^{25} column relates to the stated batch of calls on blocks of 2^{22}, 2^{22}, 2^{23} and 2^{24} points.

Authors' response

Section 4.3 now describes how the size ladders are timed: sizes up to 2^{21} as single calls, larger sizes from one pass over consecutive blocks of 2^{22}, 2^{22}, 2^{23} and 2^{24} points, so that the value at $N = 2^{25}$ is the time for all four blocks divided by 2^{25}. The captions of Table 3 and Figures A3 and A4 now refer to this description.

Changes in the manuscript / artifacts: Section 4.3; captions of Table 3 and Figures A3 and A4.

ED-REC-13 RECOMMENDED

Reviewer 3: report dispersion for the headline timing tables, so a reader can judge whether a reproduced number falls within run-to-run variation.

Authors' response

See ED-REQ-01.

Changes in the manuscript / artifacts: Tables 3, 4 and B10.

ED-REC-14 RECOMMENDED

Reviewer 3 provide a map from each table and figure to the command that regenerates it.

Authors' response

Table B13 maps each generated table and figure, now including Figures 1 and A1, to the generated file and to the command (`make figures` or `make tables`) that regenerates it from the run directories of release v0.9.6. The generating code is part of release v0.9.6; the versions in the paper can differ from the generated files in caption, cropping and styling.

Changes in the manuscript / artifacts: Table B13.

1.3 Optional editorial items

ED-OPT-01 OPTIONAL

> [methods-soundness] Primary CPU results come from one AMD part with one Intel part for comparison, and GPU results from a single A100. Section 6 already notes this. A multi-architecture study (other AVX-512 microarchitectures, a consumer GPU) would show how far the routing boundaries and speedups transfer, and can be left to later work.

Authors' response

We agree that measurements on further architectures would show how far the routing boundaries and speedups transfer. This is beyond the scope of the present paper; the benchmark suite (`make reproduce`) supports such runs on other hardware.

Changes in the manuscript / artifacts: None.

ED-OPT-02 OPTIONAL

> [visualisation] In the dense scatter overviews (A10, A11, A16, A17) the median-to-maximum "arrow" spanning each series and the hatched off-scale band at the top carry a lot of information in a small space. A short worked reading of one panel in the caption, or slightly larger panels, would make these easier to parse. The paper stands without this.

Authors' response

The caption of Figure A10 now includes a worked reading of one panel; the captions of the other overview figures refer to it.

Changes in the manuscript / artifacts: Captions of Figures A10 and A16.

ED-OPT-03 OPTIONAL

> Reviewer 1: A small end-to-end demonstration inside a real GP estimation loop would settle how the pointwise gains translate to inference wall-clock.

Authors' response

Section 6 now estimates the share of covariance evaluation against the factorisation in a likelihood iteration (see ED-REC-08). We feel that a complete estimation loop, whose runtime also depends on the optimiser and the number of iterations, is beyond the scope of this paper.

Changes in the manuscript / artifacts: Section 6.

ED-OPT-04 OPTIONAL

Reviewer 2: Measurements on further CPU and GPU architectures would show how far the speedups carry beyond the EPYC 9454 and the A100.

Authors' response

See ED-OPT-01.

Changes in the manuscript / artifacts: None.

1.4 Production-proof corrections

Original editorial note: We will correct these in production, and you confirm them in the proof.

ED-PROOF-01 PRODUCTION

[writing-quality] The figure captions for A8 and A9 read "Dependence of the maximum ULP error in dependance of the covariance-assembly strategy", which both repeats "dependence/dependance" and misspells the second instance. The same "in dependance of" appears in the caption of A15. Consider "as a function of" in all three.

Authors' response

Corrected.

Changes in the manuscript / artifacts: Captions of Figures A8, A9 and A15.

ED-PROOF-02 PRODUCTION

[writing-quality] In Section 3, "$M_\nu(h) \to 0$ as $h = +\infty$" should read "as $h \to +\infty$".

Authors' response

Corrected.

Changes in the manuscript / artifacts: Section 3.

2 Reviewer 1

General assessment (original wording)

This is a careful and unusually well-documented software paper on a problem that genuinely recurs in spatial statistics and Gaussian-process work. The Matérn covariance is evaluated enormous numbers of times when a covariance matrix is assembled, and the cost grows when the smoothness parameter is estimated and the matrix is rebuilt across optimisation steps. The authors target exactly that kernel and are explicit that they do not target the full inference pipeline. The core idea, folding the complete Matérn prefactor into each numerical representation of K_ν so the intermediate $(z/2)^\nu$ and $1/\Gamma(\nu)$ factors never separately under- or overflow, is sound and directly addresses a failure mode that practitioners hit in real parameter ranges.

Authors' response

We are grateful for this summary, which captures the motivation of the work well. We have kept the explicit restriction to the kernel and, following the suggestions below, added an estimate of its share in a likelihood evaluation (Section 6).

The strongest application result is not the speed, it is the robustness. Table B6 shows that the Bessels.jl and Boost ports produce non-finite covariances on about 12% of the small-z set and about 32% of the large-ν set, while libmatern returns finite values throughout. A spatial model with large smoothness, or with sites at small separation relative to the scale, would receive NaN or Inf covariances from standard libraries, and a user would then be debugging a factorisation failure with no obvious cause. This result deserves a sentence in the abstract and the introduction rather than its current place in an appendix table. It is the finding most likely to change what a practitioner installs.

Authors' response

We agree that this is an important practical result. Section 5.2 now reports the shares of non-finite results and their consequence for covariance matrices; on its placement in the abstract, see R1-2.

The performance claims are honestly hedged, and I want the hedging carried into the headline numbers. The $6.6\times$ CPU figure is a maximum at the largest batch, measured on the random mixture that Table B5 describes as chosen to exercise routing regions, with the explicit note that it was not fitted to observed covariance parameters. An application reader scanning the abstract will read $6.6\times$ as the gain to expect, when the common fixed-half-integer case ($\nu = 2.5$, so Matérn-5/2) runs on the dedicated polynomial path with a far larger factor, and a single-region workload will differ again. Please state a typical value next to the maximum and name the benchmark nature in the abstract. On the GPU side the paper already does the right thing by reporting the end-to-end median of $1.1\times$, which tells the reader plainly that once host-device transfer is counted the practical GPU benefit over existing GPU libraries is small and the real GPU value is kernel accuracy and finiteness. That candour is welcome.

Authors' response

We have carried the qualification into the headline: the abstract now states $6.6\times$ as the speedup for the synthetic random mixture at $N = 2^{25}$ together with the range and median over the eight datasets, and Section 5.3 adds the value for $\nu = 2.5$ (R1-1).

On accuracy, the per-dataset picture in Table 2 is more textured than "competitive or improved" suggests at a glance. On the grid and random sets, which are the broad coverage sets closest to a general parameter sweep, Boost reaches lower median and maximum errors than the AVX-512 backend. The paper states this in Section 5.2, so there is no inconsistency, and I note it only so the framing matches the evidence: the accuracy advantage is clearest in the stress regimes (small z, large ν) and in the (h, ν, s) interface, while in the broad regimes accuracy is competitive and the win is throughput and finiteness. The (h, ν, s) analysis in Section 6 is, from an application standpoint, the most useful comparison in the paper, because users pass physical distance, scale

and smoothness, not a pre-rounded z, and the argument that external libraries lose accuracy to that rounding is a real interface point. It should not be buried at the end.

Authors' response

We agree with this reading. Section 5.2 now states on which datasets Boost achieves smaller errors, and the (h, ν, s) comparison has been moved from the Discussion to Section 5.2 (R1-3).

The one substantive gap for my role is the step from pointwise covariance accuracy to inference. The authors say clearly that pointwise accuracy need not imply stability in covariance-matrix factorisation or in the downstream likelihood, and they flag it for future work. I accept that scoping. The paper would still be more convincing to its intended audience with a short, cheap bridge using the material already present: Table B11 already counts the evaluations needed to build a matrix, so a back-of-envelope of what fraction of a varying-ν estimation loop those evaluations occupy would let a reader judge where a $6.6\times$ (or $72\times$) kernel gain actually lands in end-to-end time. A full GP demonstration is more than this paper needs and I mark it optional.

Authors' response

We thank the reviewer for accepting the scoping. Section 6 now gives the suggested estimate based on Table B11 (R1-4); a complete estimation loop remains beyond the scope of this paper (R1-5).

Numbered suggestions to the authors

R1-1 REQUIRED

In the abstract and in Section 5.3, report a typical CPU speedup (for example a median across datasets or at a representative batch size) next to the maximum $6.6\times$, and state that the random benchmark mixture (Table B5) is constructed to cover routing regions, so the headline figure is a benchmark maximum and not a workflow expectation.

Authors' response

See ED-REQ-02.

Changes in the manuscript / artifacts: Abstract; Section 5.3.

R1-2 RECOMMENDED

Move the finiteness result forward: state in the abstract and introduction that common libraries return roughly 12% non-finite values on the small-z set and 32% on the large-ν set (Table B6), while libmatern returns none, and explain the application consequence for covariance-matrix assembly.

Authors' response

See ED-REC-06.

Changes in the manuscript / artifacts: Section 5.2.

R1-3 RECOMMENDED

Promote the (h, ν, s) interface comparison from Section 6 to the main results, and make clear there that the headline (z, ν) numbers compare numerical methods while (h, ν, s) compares the interface a practitioner actually uses.

Authors' response

See ED-REC-07.

Changes in the manuscript / artifacts: Section 5.2.

R1-4 RECOMMENDED

Add a short estimate, built on Table B11, of the share of a likelihood-estimation loop (with ν estimated, so the matrix is rebuilt each step) that covariance evaluation occupies, so readers can place the kernel speedups in inference wall-clock terms.

Authors' response

See ED-REC-08.

Changes in the manuscript / artifacts: Section 6.

R1-5 OPTIONAL

An end-to-end demonstration inside a real Gaussian-process estimation loop, reporting the covariance-assembly share against factorisation, would settle the practical-benefit question for the paper's intended audience.

Authors' response

See ED-OPT-03.

Changes in the manuscript / artifacts: Section 6.

3 Reviewer 2

General assessment (original wording)

This is a careful numerical paper. The core methodological idea, folding the full Matérn prefactor into the series and asymptotic representations of K_ν so that the intermediate $(z/2)^\nu$ and $1/\Gamma(\nu)$ factors never form on their own, is well motivated by the underflow and overflow behaviour set out in Section 2.1, and the derivations in Sections 2.2 and 2.3 are presented in enough detail to follow. The accuracy methodology is the strongest part. High-precision references from FLINT/Arb enclosed to at least 110 bits, stored as two binary64 components, with the ULP metric of Equation 35 and an explicit check that rlo quantisation below 2^{-972} does not move the reported maxima, is the right way to measure these errors. The treatment of the baselines is fair in a way that works against the authors' own headline: three assembly strategies per external library, the finite-result-then-lowest-max-ULP selection rule, and separately prepared modified variants of CuBesselK and Boost. That is the conservative choice and it is the correct one.

Authors' response

We appreciate the close reading of the derivations and of the accuracy methodology. We are glad that the reference computation, the ULP metric and the conservative treatment of the baselines are considered appropriate; we have kept them unchanged and, as suggested below, applied the same care to the reporting of the timings.

My reservations are about the reporting of the timing results rather than the numerical methods. The timing protocol gives two warm-ups and 25 repetitions on an exclusive node with an invariant TSC, but the text does not say which statistic over those 25 repetitions becomes the per-element cost in Tables 3, 4 and B10, nor does it report any dispersion. For a benchmarking paper the summary statistic (median, minimum, mean) and its spread are part of the claim, because a 6.6x ratio formed from two unstated point estimates cannot be read for stability. Please state the statistic and add a spread, which the 25 repetitions already contain.

The abstract leads with a maximum speedup. A maximum over the size ladder is the single most favourable operating point, and the per-dataset tables show the end-to-end GPU picture is much flatter (median 1.1x, range 1.00x to 1.81x), which the paper reports plainly. Carrying the same honesty into the CPU headline, by giving the median or the distribution of the speedup across ladder sizes next to the maximum, would make the summary match the body.

Authors' response

We have followed both suggestions. The per-element costs are now stated as medians of 25 repetitions with interquartile ranges, and the speedups are given with bounds from the quartiles of both implementations (R2-1); Section 5.3 reports the median and range over the ladder sizes next to the value at $N = 2^{25}$ (R2-2).

Two further items affect whether a reader can reconstruct the numbers. Compiler optimisation flags and FMA contraction settings change both accuracy and runtime, and the distinction between matched and matched_fma shows the authors know this, yet the flags themselves are not stated in the text. And since the ULP tables are exact down to single points, run-to-run determinism of the SIMD and GPU reductions is part of what is being claimed, so a short determinism attestation plus one reproduction command would let the journal verify the tables against the archived data. The environment in Table B7 already covers most of what is needed.

Authors' response

Section 4.4 and Table B7 now state the compiler flags and contraction settings, Section 4.4 attests determinism and reports a repeat of all runs, and Table B13 maps the tables and figures to the commands that regenerate them (R2-3, R2-4).

The scope limits are stated well in Section 6: pointwise accuracy rather than full Gaussian-process

pipelines, and benchmark datasets that are not claimed to match application parameter distributions. The one scope point I would add is hardware. The speedups rest on one primary CPU and one GPU, and how they transfer to other SIMD widths and GPU generations is open. That is beyond this paper and is marked optional.

Authors' response

We agree that measurements on further hardware would be valuable and leave them to later work (R2-6).

Numbered suggestions to the authors

R2-1 REQUIRED

State which summary statistic over the 25 repetitions the per-element costs in Tables 3, 4 and B10 report, and add a dispersion measure (for example an interquartile range or a min-to-max band) for the reported speedups.

Authors' response

See ED-REQ-01.

Changes in the manuscript / artifacts: Section 4.4; Tables 3, 4 and B10.

R2-2 RECOMMENDED

Report the median or the distribution of the CPU speedup across the size ladder next to the maximum, so the abstract headline reflects typical behaviour and not only the best operating point.

Authors' response

See ED-REC-09.

Changes in the manuscript / artifacts: Section 5.3.

R2-3 RECOMMENDED

State the compiler optimisation flags and the FMA-contraction setting used for the accuracy builds and the timing builds.

Authors' response

See ED-REC-10.

Changes in the manuscript / artifacts: Section 4.4; Table B7.

R2-4 RECOMMENDED

Add a determinism attestation for the SIMD and GPU reduction orders and a single command that regenerates the benchmark tables from the archived arrays, with a pointer to the captured environment.

Authors' response

See ED-REC-11.

Changes in the manuscript / artifacts: Section 4.4; Software and data availability; Table B13.

R2-5 RECOMMENDED

Clarify in the Table 3 caption how the N=2^{25} column relates to the stated sequence of calls on blocks of 2^{22}, 2^{22}, 2^{23} and 2^{24} points.

Authors' response

See ED-REC-12.

Changes in the manuscript / artifacts: Section 4.3; captions of Table 3 and Figures A3 and A4.

R2-6 OPTIONAL

Measurements on additional CPU and GPU architectures would show how far the reported speedups extend beyond the EPYC 9454 and the A100.

Authors' response

See ED-OPT-01.

Changes in the manuscript / artifacts: None.

4 Reviewer 3

General assessment (original wording)

This is a careful piece of numerical software work, and from a reproducibility standpoint it starts from a strong position. The inputs are fixed binary64 arrays archived on Zenodo, the high-precision references are computed with FLINT/Arb to at least 110 bits and stored as two binary64 components, and the accuracy metric is defined explicitly in Equation (35) down to the nextafter call. Because the benchmark points are frozen and the references are archived, the accuracy tables are in principle exactly reproducible, which is more than most benchmarking papers in this area offer. The baseline handling is also unusually fair: versions are named, the K_ν-to-Matérn assembly is selected per dataset to avoid handicapping external libraries, and the non-finite counts are tabulated rather than hidden. The end-to-end GPU result of $1.1\times$ median, set against kernel speedups up to $6.8\times$, is reported plainly, and Table B6 and Figures A8 and A9 make the failure behaviour of competitors legible. I did not run a reproduction in this pass, so the comments below concern what the manuscript and the declared artifacts let a reader verify.

Authors' response

We are pleased that the reviewer finds the archived inputs, references and baseline handling a strong starting point, and we appreciate the focus on what a reader can verify from the declared artifacts. Building on these, the revision adds the build configuration, the pinned upstream versions, a repeat of all runs and a map from the tables and figures to the commands that regenerate them, so that the timings can be checked in the same way as the accuracy tables.

My main reproducibility concern is the build environment behind the ULP numbers. ULP distances for SIMD and FMA kernels are sensitive to compiler optimisation level, target architecture flags, and floating-point contraction settings. Table B7 gives compilers, toolkit, driver and hardware, which is good, and the matched and matched_fma backends show you are aware that FMA changes the result. What I cannot find is the exact flag set (optimisation level, -march, the contraction policy) that the reported numbers were produced under. For a software paper that ships code with its results, a captured and pinned build configuration plus a short attestation that a repeat run reproduces the reported ULP and timing values is the reproducibility floor. If the repository already contains the build scripts, the ask reduces to naming them in the paper and stating the determinism result.

A second verifiability gap is specific and the manuscript names it itself: the kaust-modified results depend on upstream CuBesselK source that is downloaded separately and not redistributed, with changes applied by a script. Without a pinned version or commit for that upstream, and for the other baselines, the modified-baseline comparisons drift the moment upstream moves. Recording the exact commit hashes closes this and costs almost nothing.

Authors' response

We have closed both gaps. Section 4.4 and Table B7 now record the build configuration of every backend, and a repeat of all runs at the same commit is reported in Section 4.4 (R3-1); Section 4.2 and Table B12 pin the upstream versions and commits of all baselines, including CuBesselK (R3-2).

Third, the headline framing. The abstract leads with "maximum $6.6\times$ speedup", and the CPU result in Section 5.3 is likewise a maximum at a single N and workload. A maximum is the least reproducible summary a benchmark can report, since it is the most sensitive to run-to-run noise and to the choice of comparison point. Please state what the maximum is taken over and report the representative CPU speedup next to it, or qualify the sentence to say it is a best case. The GPU tables already give per-dataset figures and a median, which is the right pattern. Relatedly, Tables 3 and 4 give point estimates with no spread, although the timing protocol uses 25 repetitions. Reporting an interquartile range or a min and max for the headline rows would let a reader decide whether a reproduced value is consistent with yours, and the box plots in Figures A5 and

A12 show you already have the machinery to do it.

Authors' response

The word "maximum" has been removed; the abstract and Section 5.3 state what $6.6\times$ refers to and give representative values (R3-3), and the timing tables report interquartile ranges (R3-4).

None of this touches the correctness of the mathematics or the honesty of the reporting, both of which read well. The asks are about letting a second person stand up the same numbers from the artifacts you have.

Authors' response

We thank the reviewer for this assessment. Table B13 (R3-5) and the archived runs are intended to make exactly this possible.

Numbered suggestions to the authors

R3-1 REQUIRED

Record in the paper, or name the file in the repository that records, the exact build configuration the ULP and timing results were produced under, including optimisation level, target-architecture flags and the floating-point contraction policy for each backend. Add one sentence attesting that a repeat run reproduces the reported ULP values and timings on the stated hardware.

Authors' response

See ED-REQ-04.

Changes in the manuscript / artifacts: Section 4.4; Table B7.

R3-2 REQUIRED

Pin the exact version or commit hash of the CuBesselK upstream that the modification script targets, given that the source is not redistributed, and likewise pin the exact releases or commits of Boost, GSL, libstdc++ and both Bessels.jl ports used.

Authors' response

See ED-REQ-05.

Changes in the manuscript / artifacts: Section 4.2; Table B12; bibliography.

R3-3 REQUIRED

State what population the "maximum $6.6\times$" CPU speedup is taken over, and report the representative (for example median) CPU speedup alongside it, or qualify the abstract and Section 5.3 sentence to present it as a best-case figure.

Authors' response

See ED-REQ-06.

Changes in the manuscript / artifacts: Abstract; Section 5.3.

R3-4 RECOMMENDED

Add a dispersion measure (interquartile range, or min and max over the 25 repetitions) to the headline timing rows in Tables 3 and 4, so a reader can place a reproduced timing against your run-to-run spread.

Authors' response

See ED-REQ-01.

Changes in the manuscript / artifacts: Tables 3, 4 and B10.

R3-5 RECOMMENDED

Provide a short table or script mapping each manuscript table and figure to the exact command that regenerates it from the archived inputs.

Authors' response

See ED-REC-14.

Changes in the manuscript / artifacts: Table B13.

R3-6 OPTIONAL

A run on a second GPU and a third CPU microarchitecture would indicate how far the empirically chosen routing boundaries and the reported speedups transfer beyond the two tested machines.

Authors' response

See ED-OPT-01.

Changes in the manuscript / artifacts: None.