Response to Reviewer

Summary
The manuscript has improved substantially in response to the previous round of reviews, and the revisions address several important concerns. The additional benchmarking and clarification of the GPU implementation strengthen the paper considerably. Subject to the points raised below, the manuscript appears close to being suitable for publication.

R:> We thank the reviewer for their positive assessment of our manuscript and their constructive feedback. As requested, we have addressed all questions and comments regarding the code's efficiency, parallelization strategies, and the verification results presented in the study.

Primary suggestions

1) **Citation of the user manual**

The manuscript repeatedly references a user manual and technical documentation, but these are not formally cited. It would improve clarity and reproducibility if this documentation were referenced consistently.

Ideally, the manual should be archived with a DOI (e.g., Zenodo) and cited in the reference list. At minimum, a stable URL citation should be provided and referenced consistently throughout the manuscript.

R:> *We thank the reviewer for this observation. We agree that consistent formal citation of the user manual and technical documentation is important for reproducibility and clarity. The documentation has now been archived on Zenodo with a DOI and is formally cited in the manuscript as:*

Meza, J., Kusanovic, D. S., Juan, F., and Meneses, R.: SWEpy: Shallow Water Equation Python solver - User Manual & Technical Reference, https://doi.org/10.5281/zenodo.19323355, 2026

2) **Interpretation of CPU vs GPU speedups**

The performance comparison between the CPU and GPU implementations requires clarification. The manuscript states that the CPU implementation is not optimized and lacks parallelism (e.g., OpenMP or MPI). However, as currently presented, the results may be interpreted as comparing the GPU implementation

against a CPU baseline whose scaling behavior differs substantially from that of a typical optimized CPU solver.

A simple scaling estimate suggests that the CPU code exhibits significantly worse-than-linear scaling with problem size. For example, using the Conical Island benchmark:

45,924 / 2,958 × 0.076 ≈ 1.17 s (expected), reported 2.58 s

This suggests that the CPU implementation scales poorly with increasing problem size. If the CPU code scaled approximately linearly, the resulting GPU speedup would be roughly:

1.17 / 1.04 ≈ 1.1

which nearly eliminates the reported advantage.

A similar effect appears in the Gaussian bump benchmark. Using the coarse case as a baseline:

1,184,128 / 18,992 × 0.73 ≈ 45.5 s (expected), reported 219 s

If the CPU code scaled ideally, the GPU speedup would be approximately:

45.5 / 8.37 ≈ 5.4

rather than the reported 26×.

This does not diminish the usefulness of the GPU implementation; a factor of ~5 improvement on consumer hardware would still be significant. However, the presentation of the results may give the impression that the GPU implementation is being compared against a CPU baseline that does not scale well with problem size.

It would strengthen the manuscript to either:

- explicitly acknowledge this limitation, or present the results primarily in terms of GPU throughput and scaling, rather than speedup relative to the CPU code.

R:> We thank the reviewer for this insightful observation. The scaling analysis presented is correct: the CPU implementation does not scale linearly with problem size. We fully agree that this asymmetry deserves explicit acknowledgment in the manuscript.

The root cause of the sublinear CPU scaling is a deliberate design decision. One of SWEpy's primary objectives is user accessibility, both in terms of usage and source code readability. Accordingly, version 1.0 was developed with a straightforward implementation mindset: calculations are written explicitly as a sequence of Python/NumPy operations rather than through fused or compiled pipelines. This is not incidental; most algorithmic decisions were made with GPU parallelism in mind, so that the same explicit, operation-by-operation code that limits CPU performance is precisely what CuPy can parallelize massively on the GPU. The CPU implementation is therefore not a performance-optimized baseline – it is a structurally equivalent, readable counterpart designed to make the GPU implementation transparent and verifiable.

This design choice leads to two specific contributing factors for the worse-than-linear CPU scaling: (1) both implementations follow a deliberately straightforward computational structure to preserve user accessibility; and (2) the wet/dry algorithm relies on comparison and indexing operations (np.where, np.count_nonzero, and fancy indexing) over arrays that may not be in contiguous memory layout due to transposition and stacking – resolving this would require restructuring the code in ways that would break the structural similarity between the CPU and GPU versions, making the comparison less fair.

We have revised Section 5.1.4 of the manuscript to make this context explicit, noting that the reported speedup values reflect the advantage of the GPU implementation over this specific unoptimized serial baseline, and should be interpreted accordingly.

Secondary comments

Note that line numbers refer to the version of the manuscript showing tracked changes.

3) **CuPy implementation:**

While a detailed explanation of CuPy internals is not necessary, it would be helpful to briefly discuss the design decisions associated with using CuPy. For example:
- how array layouts and memory transfers were handled
- whether kernel fusion or custom kernels were considered
- limitations encountered when mapping the algorithm to CuPy

These points could also be addressed in the user manual if preferred.

R:> We thank the reviewer for their suggestion. A brief comment was added to the results chapter to touch on these points, which are also reviewed in Chapter 4 of the user manual.

4) **(Table 1)** The description of GeoClaw could be clarified.

GeoClaw has GPU-related work and primarily uses OpenMP for shared-memory parallelism. MPI support is also available in the dispersive version through PETSc.

R:> We thank the reviewer for this correction. We have updated the GeoClaw entry in Table 1 to more accurately reflect its current parallelization capabilities. The Parallelization framework column now reads "OpenMP" instead of "MPI," as OpenMP is the primary shared-memory parallelization strategy used in GeoClaw for patch-based AMR computations (https://www.clawpack.org/openmp.html). The *GPU* column has been changed from "No" to "Partial†," with a footnote indicating that GPU versions of GeoClaw have been developed (Qin et al., 2019; https://doi.org/10.1029/2019MS001635) but have not yet been merged into the main Clawpack release (http://www.clawpack.org/gpu.html). The footnote also notes that MPI support is available in the dispersive (Serre–Green–Naghdi) version through PETSc.

5) **(Table 6)** The addition of Table 6 is very helpful and significantly improves the presentation of the results.

R:> We thank the reviewer for this positive feedback. We are glad that the addition of Table 6 has improved the clarity and presentation of the computational performance analysis.

6) **(line 123)** The phrase “(e.g. infiltration/transport)” appears misplaced and may require editing.

R:> We thank the reviewer for this observation. The parenthetical example "(e.g., infiltration/transport)" was indeed misplaced in the original text, interrupting the description of the GPL licensing rationale. We have removed it from the Introduction (line 107) and relocated a revised version “(e.g., infiltration, sediment transport)" to Section 4, where it now accompanies the discussion of the modular architecture's extensibility. This placement more clearly conveys that these are examples of physical processes that could be incorporated through the framework's modular design.

7) **Forward Euler notation:** The change from “FE” to “EE” (explicit Euler) may not improve clarity. Once FE is established, it may be preferable to retain that abbreviation and clarify it in the figure captions.

R:> We thank the reviewer for this suggestion. After careful consideration, we decided to adopt neither "FE" nor "EE" as an abbreviation, and instead use the full term "Explicit Euler" consistently throughout the manuscript, including figure captions. We believe that writing the method name in full avoids any potential ambiguity associated with abbreviations and improves readability for a broader audience. This change has been applied consistently across all relevant sections and figures.

8) **Error metrics:** The response suggests that pointwise comparisons may be misleading. While this is true, there are well-established metrics that can better capture differences in wave fields. For example, metrics based on Wasserstein distances or other distribution-based norms may better characterize the spatial structure of the error. Although the current validation results are likely sufficient, the manuscript could benefit from briefly discussing alternative metrics or explaining why the chosen metrics are appropriate.

R:> The referee raises a valid point regarding the use of alternative error metrics. Indeed, distribution-based measures, such as those based on Wasserstein distances, can provide additional insight into the spatial structure of the error, particularly in problems where transport effects play a dominant role.

However, the present work focuses on several aspects related to the development and validation of the numerical results, and the comparisons are primarily conducted either pointwise or through spatial norms. In this context, the L^2 norm provides a direct and meaningful measure of the discrepancy between the numerical and reference solutions, especially given that the validation is performed against experimental measurements.

Based on our study, we would like to emphasize the following:

- In Table 2, we compare the flow variable and the reference solution in terms of their spatial distribution using the L^2 norm. Since the objective is to assess the method's ability to reproduce the system's stationary state, this metric is appropriate, as it directly quantifies pointwise discrepancies. In this stationary setting, where no dynamic transport effects are considered, a pointwise measure such as the L^2 norm is sufficient to characterize the quality of the approximation.
- In Tables 3 and 4, comparisons are presented at specific measurement points. Therefore, we consider that the use of the L^2 norm allows us to draw meaningful conclusions regarding the characteristics of the numerical results, particularly since these comparisons involve experimental data.
- We note that, while Wasserstein-type metrics can in principle be extended to two-dimensional problems such as the conical island benchmark, their computation requires solving a high-dimensional optimal transport problem, which is significantly more expensive than standard norm-based

comparisons. In practice, such approaches often rely on regularized formulations (e.g., Sinkhorn iterations) or dimensional reductions.

- For the conical island problem considered in this work, we have also employed an energy-based measure to capture information about the development of numerical diffusion. This metric, together with the comparisons at measurement stations, provides sufficient evidence to support the conclusions of the present study.

We appreciate the reviewer's suggestion to consider more sophisticated metrics such as Wasserstein distances. This is a genuinely interesting direction that could enrich future validation studies, particularly for problems where spatial transport structure plays a more central role. We hope the clarifications above demonstrate that the current choice of metrics is well-justified within the scope of this work, and we welcome this as a valuable avenue for the future development of the framework.

9) **Scaling discussion:** Strong vs. weak scaling is currently hinted at in the GPU discussion, and clarifying this distinction may further strengthen the performance analysis.

R:> We thank the reviewer for this suggestion. In the revised Section 5.1.4, the performance analysis already describes the transition from latency-bound to throughput-bound execution as problem size increases on a single GPU. To further clarify, we have added a closing paragraph that explicitly distinguishes this problem-size scaling analysis from formal strong and weak scaling studies, which evaluate performance as the number of processing units varies. Since SWEpy currently targets single-GPU execution on consumer-grade hardware, our analysis instead focuses on characterizing how efficiently the solver utilizes the device across a range of mesh sizes. We also note that a multi-device scaling study remains a relevant direction for future work.

10) **GPU utilization table:** The GPU utilization table is informative, but not as interpretable as it could be. Clarification of (1) how utilization was measured and (2) whether it represents SM occupancy, kernel throughput, or another metric would help readers interpret these results.

R:> Regarding GPU throughput: the paragraph preceding Table 6 now explicitly defines GPU throughput as SM occupancy, specifically, the average percentage of warps active in streaming multiprocessors during runtime, as derived from system traces. This definition is provided inline when the table is introduced, so the reader does not need to consult external documentation to interpret the values.

Regarding VRAM utilization: the paragraph following Table 6 now clarifies that memory usage is reported as the ratio of allocated to total available memory, and discusses how this metric behaves across problem sizes. In particular, it explains why the fine-resolution Conical Island grid exhibits higher memory utilization than the coarser Gaussian Bump grid despite having fewer elements – a result of the additional memory demands imposed by the active wet/dry handling algorithm.

Together, these additions ensure that Table 6 is self-contained and interpretable without ambiguity, in line with the reviewer's suggestion.