Reviewer #2

First of all, I acknowledge the work of the authors to further merge data- and physics-based approaches to modeling by showing how models using implicit solvers can be integrated into a typical machine-learning workflow with backpropagation at its core. This is a valuable contribution to the hydrological modeling sciences, but unfortunately the authors do not convincingly prove in their paper the immediate benefit thereof, and they obscure their point by adding aspects to the study that are not related to the main message. I will explain this in the following:

A) The authors are correct that mainstream conceptual hydrological models like HBV have traditionally - and often without much reflection – been used with simple explicit numerical schemes, and a pre-set order of process execution, and that this may cause substantial problems (see Clark and Kavetski, 2010 as cited by the authors), and that implicit schemes can solve these problems. Therefore, in this manuscript, in addition to the description of how to include implicit schemes in MLworkflows, I was expecting a demonstration of how this actually solves a problem. That is, showing that for a particular hydrological modeling task (here: modeling streamflow in daily resolution of the CAMELS-US basins) i) the standard explicit scheme introduces problems and ii) that an implicit scheme solves them. The authors mention this point in the paper (line 324-326), but unfortunately do not address it. For example, one could operate HBV models for some of the CAMELS catchments with various execution orders and extremely fine-grained time-stepping, thus effectively removing the detrimental effect of the explicit scheme, and then compare to a standard time stepping and execution order, and to a model using an implicit scheme. The authors conclude in their study that the (small) model improvements between the HVB-hybrid variants using explicit and implicit schemes are due to problems introduced by the explicit scheme (lines 412-415), but because they do not provide a proof for a cause, the conclusion based on an effect is not convincing. In this context, it might also be interesting to analyze if decreasing negative effects of explicit schemes by higher time stepping (or other changes to the model computational setup) might be more efficient than shifting to implicit schemes. The authors mention that computational costs for the latter increased by a factor of 5-10 (line 581). Increasing the time stepping of the explicit scheme from daily to 6 hours would only mean a factor of 4, but would already resolve diurnal cycles, which might be relevant additional information for the model.

Thank you for your suggestions. The reviewer's main point is that one can use adaptive or much smaller time steps with explicit schemes so that implicit schemes no longer have an advantage. Well, yes and No. We have several points of response:

(1) Yes, you can reduce time steps, but with automatic differentiation (AD), each step (especially those with thresholds) incur memory usage, CPU overhead and add to the length of the gradient chain, in addition to adding to the computational expenses during forward. Many times we need threshold functions even with a small time step because some operations like logarithm cannot admit zero or the smallest negative values. Using a small time step will incur more memory use.

(2) We agree that explicit schemes are valuable can be used in many cases, but it has been studied extensively in numerical algorithms that stiff ODEs are best handled by implicit schemes (Sundnes, 2023, https://en.wikipedia.org/wiki/Stiff_equation). These numerics are

documented in many decades of literature and we believe it no longer requires us to prove it. Here we are saying we must enable implicit solvers, not saying explicit solvers cannot be used. They both have their advantages and disadvantages.

(3) Batch-dimension parallelism is absolutely crucial because the point of differentiable modeling is to support big-data learning. Batch enables learning across many basins or instances, but the solver may also run into different numerical characteristics and time-stepping requirements, rather than the uniform operations preferred by the GPU. This is why adaptive time stepping is tricky for running differentiable models with minibatch and the adjoint solves an important problem. In fact, we have tried adaptive explicit ODE solvers, and while they work beautifully for one problem, they do not work well for parallel simulations with a batch. Furthermore, to use small time steps rigorously, in theory you need to match forcing inputs to those tiny steps, which requires interpolation schemes and potentially adds lots of complexity.

We will add these points into the following paragraph to the revised paper:

"*While this paper focuses on enabling implicit solvers in differentiable modeling, we do not suggest that explicit solvers are to be discouraged. Runge-Kutta schemes can be well suited for a number of cases and may be attempted for the rainfall-runoff case. It has long been explored in the numerical algorithm literature that each type of solvers has their advantages and disadvantages and is suitable for different problems. For example, implicit solvers are not only preferred but also necessary for stiff ODEs, especially those with dynamics on vastly different time scales and those resulting from the discretization of elliptic PDEs. Using explicit solvers for them could necessitate very small time steps which need to be coordinated with the modification of forcing inputs. In the context of differentiable modeling, a new dimension of consideration plays an important role --- GPU parallel efficiency at the batch level --- because the primary point of differentiable modeling is to learn from big data. Either explicit or implicit scheme needs to serve this purpose. This means that time-adaptive solvers that may require vastly different time steps amongst batch members may have limited applicability when we want to use minibatches. In addition, as discussed in the Introduction, all automatic differentiation steps incurs CPU overhead and storage burdens --- thresholds and array mutation, especially, often require data storage on the GPU. GPU memory may soon run out if we have too many iterations, either with explicit or implicit schemes, which could limit the training lengths. If neural network weights participate in the calculations of these iterations, it further induces the problem of vanishing gradients. We need to put these constraints into consideration and design balanced algorithms.*"

We appreciate the suggestion, given our above suggested revision that *we are here to* ***enable*** *implicit solvers but this should not discourage explicit solvers*, we believe it is unnecessary for us to run the model at extremely small time steps to prove the point. In fact, we continue to use our sequential code with the understanding that it gives us a bit higher efficiency but a bit lower numerical performance. If we run many many small time steps, we gain back numerical performance but then lose back efficiency.

In fact, as explained immediately above, running the model at extremely small time steps with AD, especially when you have operations that require data storage (you need this nonetheless as you cannot allow negative values in some operations and explicit algorithms cannot guarantee nonnegativeness), is impractical for explicit solvers due to GPU memory

usage, for the same reason many iterations pose problems for implicit schemes. It has also been shown before differentiable modeling that fixed-step explicit schemes with shorter time steps only provide a poor balance between accuracy and efficiency (Clark et al. 2010).

Running this model on a fine time step also seems not to make much sense ---- if you run on hourly or minute time scale, you are supposed to also need at least hourly inputs which requires more data preprocessing work, and will most likely run out of GPU memory before running the model for a year. It also seems that it should not be our group's responsibility to provide something like a parallel adaptive explicit solver --- in fact this could be quite hard to do: running time-adaptive solvers may also run into challenges to batch-level GPU parallel efficiency for the purpose of learning from big data. Some adaptive schemes that work well for individual basins may not work for the batch on the GPU. Running on CPU in MPI could work, but it is substantially more involved in coding and comes at 2-3 orders of sacrifice in energy cost, which most people in machine learning do not want to do.

Given these considerations, we did not implement adaptive time stepping methods initially (we actually tried this with another package in Julia but such algorithms did not support high GPU parallelism across the batch members, so we settled for pytorch and implementing our own solvers). But if not adaptive, how small a time step is enough? Tiny time steps kill the GPU ram. We shouldn't be micromanaging the time step for each different case we run. The PI here admits that the choice of these solvers and algorithms in fact resulted from quite some elaborate exploration and messing around with various alternative Scientific Machine (SciML) packages and since 2021 during his sabbatical time, and there are many reasons why we settled on our choices.

If the editor insists that we try small time steps, we could give it an earnest attempt, but we think **it would be a little bit unfair to put this responsibility on us**, while delaying us from working on other important problems we think that need to be addressed in this new domain. We very much welcome the community to contribute to the comparisons, as there is enormous space here for the next developments. Hence, while we very much appreciate the constructive opinions, we respectfully disagree with the reject recommendation. We suggest that the above two issues raised by Dr. Ehret could be addressed by revising the manuscript, making clarifications and stating limitations, as the paragraph proposed above.

Unfortunately over the AGU and winter break time frame the interactive discussion has ended, we wonder if we could discuss more about this.

Reference:

Clark, Martyn P., and Dmitri Kavetski. "Ancient numerical daemons of conceptual hydrological modeling: 1. Fidelity and efficiency of time stepping schemes." *Water Resources Research* 46, no. 10 (2010).

Kavetski, D. and Clark, M.P., 2010. Ancient numerical daemons of conceptual hydrological modeling: 2. Impact of time stepping schemes on model analysis and prediction. *Water Resources Research*, *46*(10).

Sundnes, J., 2023. Solving Ordinary Differential Equations in Python (Vol. 15). Springer Nature.

Here are our response to the detailed comments in part A from Dr. Ehret:

“Therefore, in this manuscript, in addition to the description of how to include implicit schemes in MLworkflows, I was expecting a demonstration of how this actually solves a problem.”

In Section 2.2.4, 'Backpropagation with a Coupled Neural Network and Process-Based Model', and Section 2.2.5, 'Adjoint-Based Implicit Scheme', we demonstrate step-by-step how implicit schemes are derived and function within differentiable models. This functionality is primarily numerical and can only be evidenced through changes in model performance, in contrast to model structures that have physical meanings.

“The authors conclude in their study that the (small) model improvements between the HVB-hybrid variants using explicit and implicit schemes are due to problems introduced by the explicit scheme (lines 412-415), but because they do not provide a proof for a cause, the conclusion based on an effect is not convincing.”

In this study, we conducted a rigorous comparison between implicit and explicit schemes. The structure of the HBV model, the hyperparameters used in the embedded neural network, and the datasets remained consistent across both approaches. The only variable was the numerical approximation. If there is an improvement in model performance, we believe it can be attributed to the reduction of numerical errors.

“The authors mention that computational costs for the latter increased by a factor of 5-10 (line 581). Increasing the time stepping of the explicit scheme from daily to 6 hours would only mean a factor of 4, but would already resolve diurnal cycles, which might be relevant additional information for the model.”

The computational costs are not solely due to the iteration steps in Newton's iteration but also because of the calculation of the Jacobian matrix in backpropagation. The Newton-Raphson solver can converge within an average of 3-4 iterations. Its forward computational cost is comparable to a 6-hour time-stepping scheme. However, the issue extends beyond computation cost and memory usage for storing gradients of each operation. More critically, it involves the potential for gradient vanishing or explosion – a well-known problem in machine learning – due to the accumulation of gradients with AD over all time steps and iterations in a training instance. The adjoint method for the implicit iterative solver can bypass gradient tracking in the iterations of each time step, which helps mitigate this problem.

B) Motivated by problems of the HBV model to simulate (near-)zero base flow during extended dry spells, the authors integrate a detailed study about the effect of adding an additional capillary rise process to the HBV model. This is a valid question and analysis, but it does not at all support the main argument of the paper about how and why implicit schemes can be integrated into modern hybrid modeling workflows. I therefore suggest presenting this analysis in another paper, and removing it from this one. In this context, it is interesting that the authors provide a range of possible adjustments to the HBV model to help it achieve (near-)zero flow (strategies 1-5 in lines 258-261; and lines 479-482). These adjustments touch very different physical subdomains and processes of the model, and one may wonder about the limitations of a supposed key advantage of physics-based models – realism and interpretability – if it remains mainly up to the user's preference which one is

chosen. In particular, I wonder why capillary rise from the lower subsurface should bypass the upper surface and directly connect to the surface soil moisture storage, and why the authors chose it this way. Based on the above points, my overall recommendation is that the key topic of the paper is worth publication, but also that the required changes will require time. Therefore I recommend rejecting the paper in present form, but strongly encourage a resubmission.

Yours sincerely,

Uwe Ehret

(B) Again, while we respect the opinion of Dr. Ehret, we doubt if deleting the change is the correct course of action. Many article carry more than one stories and this is a beneficial (although not that major) improvements to the model. We do not want to write another article for this change. Also, the differences between these models are very well documented ---- that is, the performance differences only from sequential to adjoint, and then from adjoint to the adjoint model with improvement structure are clearly provided in clear detail. The readers need to know what is the impact of solution accuracy, and this second story put things into context regarding how that impact compares to a change in the structure. After fixing the parameter and numerical errors, the model is now ready to understand the defects in its structure. Previously, numerical errors, parameter errors, and model structure errors were intertwined. Now, we are able to separate them and learn new physics and compare their effects. Hence, we think this is quite a useful comparison and should be retained.

We used the additional capillary rise process as an example to show how the differentiable model can be further improved with structural modifications. The current structure of capillary rise is learned from the GSFB model, which represents the recharge from deep groundwater. We think it is more appropriate to term it ‘capillary rise’ (Ye et al. 1997, Model 20 in Knoben et al. 2019). The fact that this component connects back to the surface reduces the complexity and reflects that some shallow subsurface flow (like lateral soil interflow) can indeed bypass the upper subsurface by following preferential flow paths. These are conceptual models that cannot fully take into account the spatial heterogeneity so some effective representation is needed.

Ye, W., Bates, B.C., Viney, N.R., Sivapalan, M. and Jakeman, A.J., 1997. Performance of conceptual rainfall-runoff models in low-yielding ephemeral catchments. *Water Resources Research*, *33*(1), pp.153-166.

Knoben, W.J., Freer, J.E., Fowler, K.J., Peel, M.C. and Woods, R.A., 2019. Modular Assessment of Rainfall–Runoff Models Toolbox (MARRMoT) v1. 2: An open-source, extendable framework providing implementations of 46 conceptual hydrologic models as continuous state-space formulations. *Geoscientific Model Development*, *12*(6), pp.2463-2480.