Reviewer #1

Dear authors, Dear editor,

Here is my review of the submitted work. I recommend accepting the manuscript after minor revisions.

Kindly

Ilhan Özgen-Xian

General comments and questions

1. The authors convincingly make an argument for implicit time integration. The forward Euler time stepping used in this work is indeed at a disadvantage if fixed time steps are used. However, it is not clear to me how higher order explicit time integration methods such as schemes from the explicit Runge-Kutta family (RK) would perform in comparison to the implicit one. If I understood correctly, some of the numerical issues mentioned in the manuscript might also be addressed by (adaptive) multistep schemes of this type. The advantage of RK-type schemes in this context is that the number of computations per time step is known a priori. In contrast, the Newton-Raphson iterative solver may require any number of steps until convergence. High order RK schemes, for example the standard RK45 or the adaptive RK-Fehlberg method, could also potentially benefit from the adjoint method presented in this paper to avoid excessive memory usage. Perhaps the authors can comment on this.
2. The authors mention that the Newton-Raphson solver introduces some overhead to the computation. On average, in the results shown in this paper, how many iteration steps were necessary for the solver to converge?

Thanks for your suggestions. We think these two questions can be answered together. The Newton-Raphson solver can converge in 3-4 iterations on average (We added this information in the main text). Its computational cost actually is less than the high-order explicit Runge-Kutta method, such as 4th order Runge–Kutta–Fehlberg method. RK methods could require more memory usage during the backpropagation process for gradient calculation because every step of the calculation needs to record information and store intermediate data. In standard ODE solvers and during our tests on chaotic ODE problems, RK methods sometimes run into stiff or stability issues and need to reduce time steps or change to lower-order methods, this further increases memory use and challenges to parallel efficiency. Thus, while some high-order RK methods can be useful (and we think they can be a list of options provided), they also have risks. In addition, the disadvantage of the explicit/implicit iterative solvers is not only the memory usage but also the gradient vanishing or explosion due to the gradient accumulation over all the time steps and iterations in the training instance. The adjoint for the implicit iterative solver can bypass the gradient tracking in the iterations of each time step.

Here, we do not argue that implicit solutions are the only way. We think that explicit solutions can in some cases be useful. Nevertheless, implicit methods are well known to provide important value for various problems so they must be provided as an option to differentiable models. For example, elliptic problems must be handled by implicit schemes and stiff ODEs are best handled by them, too. We will add some explanations: "*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.*"

Minor comments

1. P.2, L.70: "graphical processing units" should be "graphics processing units"

Will revise as suggested.

2. P.3, LL.105ff.: Does "elliptic operator" in this context correspond to the Laplacian? If so, some of the examples might require some annotation. The Saint-Venant equation only contains Laplacian operators if molecular/turbulent diffusion is accounted for. Many forms of the Saint-Venant equation omit these terms, for example (García-Navarro et al., 2019, doi:10.1007/s10652-018-09657-7; LeVeque et al., 2011, doi:10.1017/S0962492911000043).

We will revise it to "shallow water equations", which is a two-dimensional Saint-Venant equation considering turbulent diffusion.

3. P.3, LL.105ff. (continued) When I looked at the paper by Aboelyazeed et al. (2023) (cited by the authors), I couldn't see Laplacians in the Farquhar model equations.

Aboelyazeed et al. (2023) is an example of systems of nonlinear equations, not an elliptic operator.

4. P6, L.209: "The same forcings ... was used" should be "The same forcings ... were used"

Will revise as suggested.

5. P.12, L.335: Should it be Eq. (28) instead of Eq. (27)? May be I am misunderstanding something.

Yes, your understanding is correct. We will fix it.

6. P.14, L.398: The authors state that the mass balance preservation of the adjoint-driven NN-HBV model might be the reason behind the improved model performance. I don't understand why the mass conservation should significantly differ from the explicit sequential NN-HBV model if the hydrological process representation remains untouched. Is this related to the use of thresholds to avoid negative storages? Can the authors elaborate a bit more?

Yes, by avoiding thresholds for negative states, the implicit model can achieve better mass conservation. This impact is significant for low flows. For example, the thresholds for lower subsurface zone storage in the current HBV model can induce minimal baseflow on dry days. More importantly, the adjoint (implicit) model greatly reduces numerical errors, thus improving the model's performance. We revised the main text to: *"The advance may be attributable to HBV.adj's reduction of numerical errors, which forces the model to more accurately represent extreme values."*

7. P.24, L.580: The additional computational cost introduced by the implicit solver is quite substantial (18 h vs. 133 h), suggesting either poor convergence or large communication overhead in the implicit scheme.

We agree with the reviewer that the computational cost of the implicit solver is substantial compared to the sequential model. However, when compared with traditional models that require basin-by-basin calibration on a CPU, it is efficient for large-scale modeling. The implicit solver can converge in 3-4 iterations, and all training is conducted on a single GPU, no need for communication between nodes. The reasons it is slower than the sequential model are twofold: 1) the HBV model is called 3-4 times in each time step, whereas the sequential model only needs to call it once, and 2) the calculation of the Jacobian matrix for multiple basins, depending on the batch size, also consumes time. There are additionally some CPU overhead issues to be explored down the road.