https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

dMoT v1.0: A Differentiable Library of 36 Conceptual Hydrological Models for Large-Sample Modelling

Xin Jing[1], Jungang Luo[1], Mingxiang Yang[2], Yaoxuan Xiao[3], and Xue Yang[1]

[1]State Key Laboratory of Water Engineering Ecology and Environment in Arid Area, Xi'an University of Technology, Xi'an, Shaanxi 710048, China
[2]State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research, Beijing, China
[3]Pearl River Water Conservancy Comprehensive Technology Center of the Pearl River Water Conservancy Commission, Ministry of Water Resources, Guangzhou, China

Correspondence: Jungang Luo (jgluo@xaut.edu.cn)

Abstract. Conceptual hydrological models represent alternative hypotheses of catchment behaviour, yet their diverse numerical implementations complicate gradient-based, large-sample modelling. We present dMoT v1.0, a PyTorch-based library implementing 36 established lumped rainfall–runoff models within a common differentiable framework. Drawing on MARRMoT formulations, dMoT retains model-specific storages, fluxes, and routing processes while adopting explicit sequential state updates and GPU-parallel computation. We evaluate numerical reliability, automatic differentiation, computational efficiency, and scalability. Water-balance tests met their prescribed tolerances, and automatic-differentiation gradients agreed with finite-difference estimates under the evaluated conditions. Compiled GPU execution achieved a median 2.89-fold forward-execution speed-up over single-threaded CPU execution across the 36 models. We further demonstrate independent calibration and differentiable parameter learning across 531 CAMELS-US catchments. The two approaches achieved median model-level test scores of 0.624 and 0.612, respectively, using a numerically stabilised Kling–Gupta efficiency metric on temporally held-out data from the training catchments. These results establish dMoT as a computational framework for applying contrasting conceptual model structures under different parameter-estimation strategies. By combining a documented model library with automatic differentiation, numerical verification, and scalable execution, dMoT supports reproducible large-sample hydrological modelling and systematic comparisons of model structures and learning approaches.

1 Introduction

Conceptual rainfall–runoff models describe catchment behaviour through assumptions about water storage, partitioning and release. Established formulations such as GR4J, HBV and VIC differ in their storage configurations, flux equations and routing arrangements (Liang et al., 1994; Lindström et al., 1997; Perrin et al., 2003). Similar discharge simulations can therefore arise from different representations of hydrological processes (Beven, 2006; Kirchner, 2006). Studies applying multiple conceptual structures across large catchment samples have documented both substantial performance overlap and variation in model suitability among catchments (Knoben et al., 2020; Spieler and Schütze, 2024; Thébault et al., 2026). Model choice also

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

reflects established practice and software availability (Addor and Melsen, 2019). These findings give structural diversity a clear experimental role. Evaluating alternative process representations under consistent conditions helps establish which conclusions depend on model structure, parameter estimation or the available observations (Clark et al., 2011). Such comparisons depend on access to documented model implementations that can be applied across the same catchments and experimental protocols.

Hydrological modelling frameworks have made these structural alternatives more accessible. FUSE provides interchangeable runoff-generation formulations (Clark et al., 2008), and SUPERFLEX constructs model hypotheses from reservoirs, transfer functions and junctions (Fenicia et al., 2011; Kavetski and Fenicia, 2011). SUMMA permits controlled changes in process representations and numerical choices (Clark et al., 2015a, b), while DECIPHeR and Raven provide further configurable modelling environments (Coxon et al., 2019; Craig et al., 2020). MARRMoT established a complementary resource by bringing literature-derived lumped rainfall–runoff models into a documented MATLAB toolbox, with 46 structures in its original release and 47 in its subsequent object-oriented version (Knoben et al., 2019; Trotter et al., 2022). Together, these frameworks support systematic investigation of alternative hydrological assumptions. Their implementations differ in model organization, numerical solution and computational interfaces. Large-sample experiments that apply many established structures consequently require a common execution interface for repeated basin-parallel simulations and gradient-dependent parameter learning.

This requirement has become increasingly relevant as hydrological studies use larger and more consistently organized datasets. CAMELS-US provides meteorological forcing, discharge observations and catchment descriptors for hundreds of basins (Newman et al., 2015; Addor et al., 2017), and Caravan extends the availability of large-sample hydrological data across regions (Kratzert et al., 2023). Regionalization has long connected model parameters with catchment characteristics (Samaniego et al., 2010; Beck et al., 2016). At these scales, independent calibration (IC) entails repeated forward simulations for each model–catchment pair, while learning a shared relationship between catchment attributes and model parameters introduces a second computational demand. Differentiable parameter learning (dPL) addresses the latter by propagating gradients through a process-based model to train an attribute-to-parameter mapping across catchments (Tsai et al., 2021). Related work has combined differentiable hydrological calculations with regional learning and explicit process representations (Feng et al., 2022; Shen et al., 2023; Ouyang et al., 2025), and Python-based modelling workflows and differentiable routing have been developed for additional applications (Ouyang et al., 2026; Hascoet et al., 2026). Efficient basin-parallel simulation supports repeated calibration and multi-model evaluation, while automatic differentiation enables shared parameter mappings to be learned through the hydrological model itself. Providing both capabilities across a diverse catalogue of established conceptual structures is the computational objective considered here.

A multi-structure differentiable implementation also requires attention to numerical reliability. Conceptual models differ in their state variables, nonlinear fluxes, threshold conditions and routing memory, and their daily operations may depend on the order in which water is transferred between stores. Because automatic differentiation follows the discrete operations represented in software (Baydin et al., 2018), model-specific calculations and numerical choices, including threshold treatment (Kavetski and Kuczera, 2007), need to be documented and verified. Water-budget tests, finite-difference gradient comparisons and cross-device output checks examine complementary properties of the implementation. Forward and backward computational benchmarks further assess its efficiency and resource requirements for large-sample applications.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Here we present **dMoT v1.0**, a PyTorch library implementing 36 established lumped conceptual rainfall–runoff structures under a shared, basin-parallel interface. The collection includes models with one to six hydrological stores and a total of 271 model-specific parameter coordinates, spanning different representations of snow, interception, soil moisture, subsurface flow and routing (Fig. 2). A common wrapper manages forcing, parameters and simulation output, while each structure retains its own daily step calculations and state-update sequence (Fig. 1). The tensor-based execution interface supports basin-parallel forward simulations for IC and gradient propagation through the scored simulation period for dPL. Compiled GPU execution is available for the model-specific daily steps. Its structural and process references are drawn from MARRMoT (Knoben et al., 2019; Trotter et al., 2022). dMoT is an independent numerical implementation, with explicit sequential calculations and documented differences from the reference formulations. Both parameter-estimation procedures use the same model-specific hydrological calculations.

We evaluate the library through numerical verification, computational benchmarks and a large-sample application. The numerical verification examines water accounting, parameter gradients and CPU–GPU output consistency across the model library. The computational experiments measure forward execution under CPU, eager GPU and compiled GPU backends, as well as scaling with catchment count, simulation length and differentiation workload. Finally, all 36 structures are applied to 531 CAMELS-US catchments using IC with separately estimated basin parameters and dPL with a shared catchment-attribute-to-parameter mapping. Both approaches are assessed over a later period in catchments already represented during dPL training. This application demonstrates independent calibration and differentiable parameter learning at large-sample scale through a common set of hydrological implementations. The model definitions and numerical conventions are described in Sect. 2, the experimental protocols in Sect. 3, the numerical and application results in Sect. 4, and their implications in Sect. 5. The Appendix documents the model inventory and implementation differences, the Supplement the verification, estimation and benchmarking protocols, and the companion Technical Notebook (hereafter Notebook) the model formulations, coordinate-level records and extended benchmark tables.

2 Model description and numerical implementation

2.1 Overview of dMoT

dMoT v1.0 provides a common differentiable implementation of 36 established lumped conceptual rainfall–runoff models. The collection draws on the hydrological structures and process formulations documented in MARRMoT (Knoben et al., 2019; Trotter et al., 2022), with independent numerical implementations in PyTorch. The models span one to six hydrological stores and contain 271 model-specific free parameter coordinates in total (Fig. 2; Table A1). Their different storage configurations, flux relations and routing arrangements are retained within a common interface for simulation and parameter estimation.

Figure 1a illustrates the organisation of the library. Meteorological forcing, model parameters and initial states are supplied with an explicit catchment dimension, allowing multiple basins to be simulated simultaneously. Shared formulations are used for recurring processes, including snow accumulation and melt, interception, evapotranspiration, infiltration, subsurface transfers, baseflow and routing. Each model nevertheless retains its own states, flux dependencies and order of daily calculations. The

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

common interface standardises the organisation of inputs and simulation outputs without imposing identical internal states or process sequences across models.

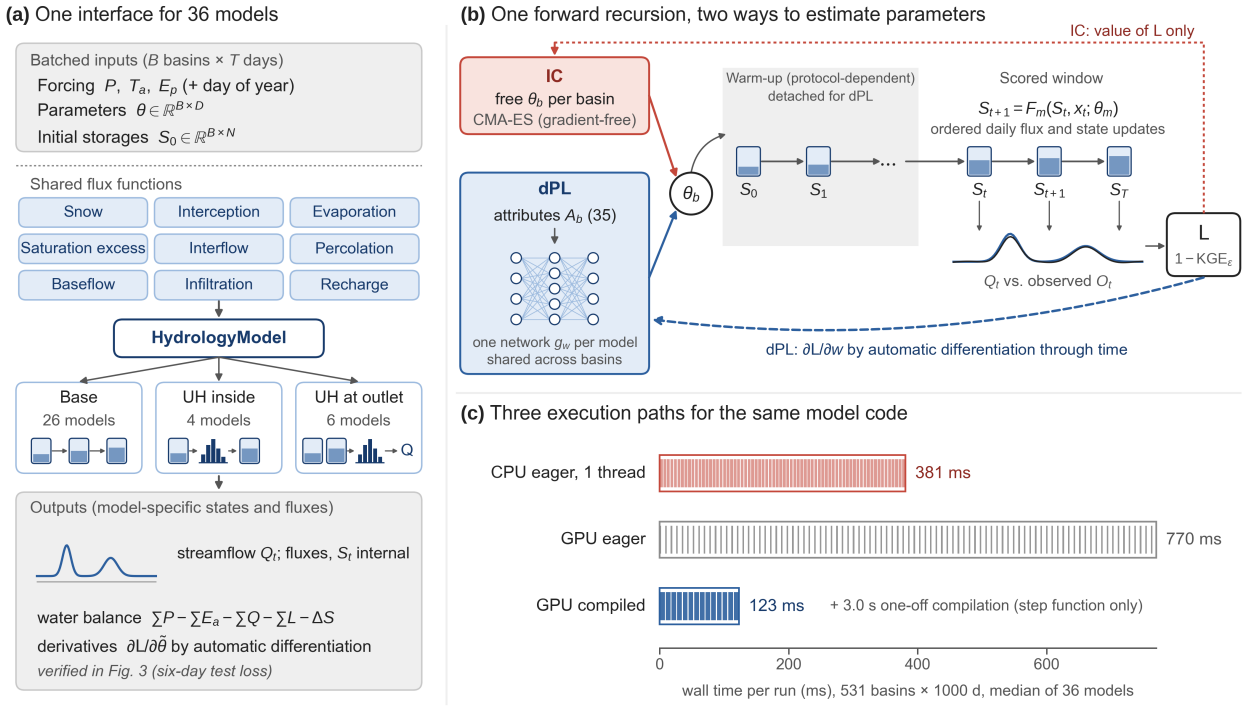


Figure 1. Architecture and computational workflows of dMoT. (a) Shared input and execution interfaces link the 36 model-specific daily calculations, grouped by the placement of unit-hydrograph routing. The water-balance sketch includes the model-declared external losses L_t of the full accounting (Eqs. 4 and 5). (b) Independent calibration (IC) and differentiable parameter learning (dPL) provide parameters to the same hydrological recursion. The warm-up block is schematic and its duration is protocol-dependent: IC repeats the first hydrological year five times (1825 d), while dPL uses 730 d in the main configuration (365 d for Penman). (c) Median forward-execution times across the 36 models for 531 catchments and 1000 simulation days under single-threaded CPU, eager GPU and compiled GPU execution; the one-time compilation overhead is shown separately.

The routing configurations in Fig. 1a comprise 26 models without a separately classified unit-hydrograph pathway, four with an internal unit hydrograph and six with outlet unit-hydrograph routing (Table A1). This classification concerns the placement of unit-hydrograph operations; the broader routing category in Fig. 2b also includes routing stores. Figure 2 presents the numbers of stores, the occurrence of major hydrological processes and the free parameter counts across all 36 models. Model identities, parameter bounds and process specifications are detailed in Sects. N2–N4 of the Notebook.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

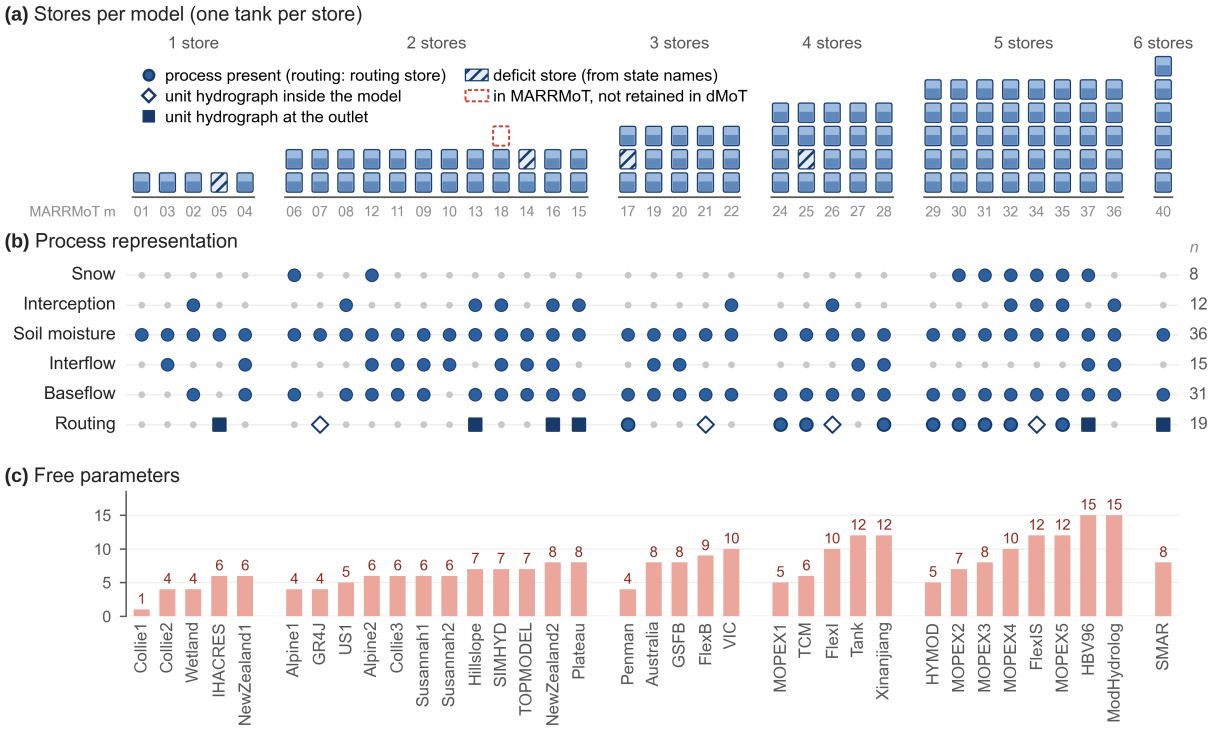


Figure 2. Structural diversity of the 36-model library. (a) Numbers of hydrological stores by model, with unit-hydrograph routing configurations indicated separately; hatched symbols denote deficit stores, and dashed red outlines indicate stores present in the corresponding MARRMoT inventory but not retained in dMoT. (b) Presence of major hydrological processes; the routing category includes routing stores and unit-hydrograph operations. (c) Numbers of model-specific free parameters.

2.2 Hydrological model representation

For model m, the hydrological state is represented by $\boldsymbol{S}_m = (S_{m,1}, \ldots, S_{m,n_m})^\mathsf{T}$, where n_m is the number of water stores and storage is expressed in millimetres of water equivalent. Meteorological inputs comprise daily precipitation P_t (mm d^{-1}), air temperature $T_{a,t}$ (°C) and potential evapotranspiration $E_{p,t}$ (mm d^{-1}), with day of year supplied where required. Each model has a vector of p_m parameters, $\boldsymbol{\theta}_m$, with structure-specific meanings and bounds. The resulting fluxes include actual evapotranspiration $E_{a,t}$, discharge Q_t and, where applicable, additional water losses L_t.

The conceptual water balance for storage i can be written as

$$\frac{\mathrm{d}S_{m,i}}{\mathrm{d}t} = \sum_{j \in \mathcal{I}_{m,i}} f_{m,j}(\boldsymbol{S}_m, \boldsymbol{x}_t; \boldsymbol{\theta}_m) - \sum_{j \in \mathcal{O}_{m,i}} f_{m,j}(\boldsymbol{S}_m, \boldsymbol{x}_t; \boldsymbol{\theta}_m), \tag{1}$$

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

where $\boldsymbol{x}_t$ denotes the forcing variables and $\mathcal{I}_{m,i}$ and $\mathcal{O}_{m,i}$ identify incoming and outgoing fluxes. The flux functions describe model-specific assumptions about water partitioning and transfer. The discrete sequence used to calculate them is described in Sect. 2.3.

Parameters are represented by normalised coordinates $\tilde{\theta}_{m,j} \in [0,1]$ (denoted $u_{m,j}$ in the Supplement) and transformed to their prescribed physical bounds $(l_{m,j}, h_{m,j})$. For IC, the transformation is linear, $\theta_{m,j} = l_{m,j} + \tilde{\theta}_{m,j}(h_{m,j} - l_{m,j})$. In dPL, the normalised coordinate is obtained from a sigmoid network output and transformed according to

$$\theta_{m,j} = \phi^{\mathrm{dPL}}_{m,j}(\tilde{\theta}_{m,j}) = \begin{cases} \exp\left[\ln l_{m,j} + \tilde{\theta}_{m,j}(\ln h_{m,j} - \ln l_{m,j})\right], & l_{m,j} > 0, \quad h_{m,j}/l_{m,j} \geq 100, \\ l_{m,j} + \tilde{\theta}_{m,j}(h_{m,j} - l_{m,j}), & \text{otherwise.} \end{cases} \tag{2}$$

Logarithmic interpolation applies to 53 of the 271 model-parameter coordinates in the reported dPL experiments (Notebook Table N3). The two estimation approaches use the same physical parameter bounds and model-specific hydrological calculations, although their mappings from normalised to physical parameters differ.

The model definitions follow the corresponding conceptual families documented by MARRMoT. Some implementations also contain substantive structural or flux-formulation differences. For example, dMoT represents SIMHYD with two hydrological stores rather than the three listed in the MARRMoT reference inventory, and IHACRES has six free parameters rather than seven. Model-specific modifications to thresholds, flux limits and selected process formulations, including those in Penman, SMAR and TCM, are documented in Sect. N4 of the Notebook. These definitions specify the dMoT formulations evaluated in this study. A concise comparison of selected implementation differences is given in Table A2.

2.3 Discrete integration, process ordering and water accounting

The production models use a sequential explicit update at the daily time step. Fluxes and intermediate states are evaluated in the order prescribed by each hydrological structure. Let $\boldsymbol{S}_t^{(k)}$ denote the state after operation k within day t. A flux-driven operation can be expressed as

$$\boldsymbol{S}_t^{(k+1)} = \boldsymbol{S}_t^{(k)} + \Delta t\, \boldsymbol{\nu}_{m,k}\, f_{m,k}(\boldsymbol{S}_t^{(k)}, \boldsymbol{x}_t; \boldsymbol{\theta}_m), \qquad \Delta t = 1\ \mathrm{d}, \tag{3}$$

where $\boldsymbol{\nu}_{m,k}$ describes the transfer of water among stores and across model boundaries. Subsequent operations may depend on storage values updated earlier in the day. The final sequence of operations gives $\boldsymbol{S}_{t+1}$ and the associated discharge, including any unit-hydrograph routing. Equation (3) describes the flux-driven updates; model-specific algebraic adjustments, bounds and routing-memory operations are incorporated where required. Accordingly, the daily calculation need not coincide with simultaneous evaluation of all fluxes in Eq. (1).

State and flux constraints are applied during these updates. Fluxes subject to available-water limits are bounded accordingly. Selected threshold processes use differentiable transitions, including a sigmoid capacity gate for saturation-excess flow (Sect. S1.4). The usual positive storage floor is 10^{-6} mm, with structure-specific zero-floor exceptions described in Sect. N2.2 and the

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

model entries of Sect. N4 of the Notebook. The precise flux sequence and numerical conventions for each model are provided in the model entries of Notebook Sect. N4.

Water accounting was evaluated at the full-simulation and daily-step scales. With $M(\boldsymbol{S})$ denoting the model-specific, signed accounting sum of water storages (including deficit-store conventions), the accumulated and daily residuals are

$$R = \sum_{t=1}^{T} P_t \Delta t - \sum_{t=1}^{T} (Q_t + E_{a,t} + L_t)\Delta t - \left[M(\boldsymbol{S}_{T+1}) - M(\boldsymbol{S}_1)\right], \tag{4}$$

$$r_t = P_t \Delta t - (Q_t + E_{a,t} + L_t)\Delta t - \left[M(\boldsymbol{S}_{t+1}) - M(\boldsymbol{S}_t)\right], \tag{5}$$

respectively, both expressed in millimetres. The corresponding acceptance tolerances were

$$a_{\text{full}} = \max(10^{-7}, n_m T\, 10^{-6})\ \text{mm},$$
$$a_{\text{step}} = \max(10^{-8}, 1.1 n_m\, 10^{-6})\ \text{mm}.$$

Figure 3a additionally uses a fixed 10^{-8} mm threshold for the accumulated residual. Daily residuals were assessed against a_{step}. Model-specific losses and accounting conventions are documented in Sect. S2.1.

2.4 Automatic differentiation and computational execution

The daily calculations are implemented in PyTorch, allowing the scored simulation period to be differentiated with respect to model parameters (Baydin et al., 2018). For each catchment, a warm-up period establishes the hydrological state before the scored interval. The warm-up states, including routing memory, are treated as fixed initial conditions for differentiation; gradients propagate through the subsequent daily state updates. Meteorological inputs, parameters and states are organised with a catchment dimension for basin-parallel execution.

The same hydrological model is used for IC and dPL (Fig. 1b). IC optimises a separate normalised parameter vector $\tilde{\boldsymbol{\theta}}_{m,b}$ for each model m and catchment b. In dPL, each model has its own neural mapping from static catchment attributes $\boldsymbol{X}_b$ to parameter coordinates. With network weights W_m, logits $\boldsymbol{z}_{m,b} = g_{W_m}(\boldsymbol{X}_b)$ and normalised coordinates $\tilde{\boldsymbol{\theta}}_{m,b} = \sigma(\boldsymbol{z}_{m,b})$, the physical parameters are $\boldsymbol{\theta}_{m,b} = \phi_m^{\text{dPL}}(\tilde{\boldsymbol{\theta}}_{m,b})$. Network weights are shared across catchments for a given model (Tsai et al., 2021; Feng et al., 2022). Gradients of the training objective $\mathcal{L}$ consequently pass through the hydrological recursion, parameter transform and network:

$$\frac{\partial \mathcal{L}}{\partial W_m} = \sum_b \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_{m,b}}\right)^{\mathsf{T}} \operatorname{diag}\left(\frac{\partial \boldsymbol{\phi}_m}{\partial \tilde{\boldsymbol{\theta}}_{m,b}}\right) \operatorname{diag}(\sigma'(\boldsymbol{z}_{m,b}))\, \frac{\partial \boldsymbol{z}_{m,b}}{\partial W_m}. \tag{6}$$

The hydrological derivative contains the dependence of discharge on preceding scored states, while the warm-up state is held fixed. The parameter-learning architecture and training protocol are given in Sect. 3.4.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

We evaluated three execution paths: single-threaded CPU, eager GPU and compiled GPU. The compiled path used PyTorch 2.8.0 to compile the model-specific daily calculation, while the temporal recursion remained outside the compiled operation (Ansel et al., 2024). All paths used double precision. Comparisons of simulated discharge are described in Sect. 3.2.

3 Experimental setup and evaluation protocols

3.1 Catchment data, study periods and objective function

The large-sample application used 531 CAMELS-US catchments (Newman et al., 2015; Addor et al., 2017), with the same catchment set and time-varying forcings applied to each model structure. The inputs comprised precipitation, air temperature and potential evapotranspiration, supplemented by day-of-year information where required. Potential evapotranspiration was estimated using the Hargreaves method (Hargreaves, 1994).

For dPL, model parameters were predicted from 35 static catchment attributes. Twenty-four attributes retained their CAMELS values, while 11 were replaced or recomputed using Caravan/HydroATLAS-derived information (Kratzert et al., 2023; Linke et al., 2019). Three positively skewed variables (catchment area, soil conductivity and geological permeability) were log-transformed, and all input attributes were standardised across the 531 catchments. Non-finite entries and categorical codes were handled as documented in Sect. S3.1 and Table B1.

The pre-test forcing record extended from 1 October 1980 to 30 September 1995. For IC, the first hydrological year was used for state warm-up, and the following 14 hydrological years (1 October 1981–30 September 1995) formed the scored calibration period. For dPL, training used 365-day scored sequences preceded by state warm-up within the pre-test record (Sect. 3.4). Evaluation covered 1 October 1995–30 September 2010, with a 365-day warm-up before the scored evaluation period. The evaluation therefore assessed a later time period at the same catchments used in dPL training.

Performance was measured using a numerically stabilised variant of Kling–Gupta efficiency (KGE; Gupta et al., 2009). For simulated discharge Q and observed discharge O over $n \geq 2$ scored days, let A_Q, A_O and C denote their unnormalised centred sums of squares and cross-products. The sample standard deviations s_Q and s_O use a variance floor $v_{\min} = 10^{-24}$ (Sect. S5.1). The performance components and resulting score were defined as

$$r_\varepsilon = \frac{C}{\sqrt{\max(A_Q A_O, v_{\min})} + \varepsilon}, \qquad \beta_\varepsilon = \frac{\overline{Q}}{\overline{O} + \varepsilon}, \qquad \gamma_\varepsilon = \frac{s_Q}{s_O + \varepsilon},$$

$$\mathrm{KGE}_\varepsilon = 1 - \sqrt{(r_\varepsilon - 1)^2 + (\beta_\varepsilon - 1)^2 + (\gamma_\varepsilon - 1)^2}, \qquad \varepsilon = 0.1. \tag{7}$$

The same KGE_ε definition was used for the reported IC and dPL scores. The dPL objective was $1 - \overline{\mathrm{KGE}_\varepsilon}$, where the bar denotes the mean over catchments with at least two valid scored days and finite discharge-series values and scores. Invalid series were excluded from that mean (Sect. S5.1). The stabilisation convention and its relation to conventional KGE are detailed in Sect. S5.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

3.2 Numerical, derivative and execution-consistency tests

Three tests examined numerical reliability: water accounting, parameter derivatives and agreement across execution backends (Fig. 3). Conservation was assessed in 188 model–forcing cases: five shared synthetic forcing cases for each of the 36 models and an additional snow-related case for eight models. The cases included dry and wet conditions, impulsive and constant inputs, and high evaporative demand, with durations of 7–1000 days. The accumulated and daily residuals in Eqs. (4) and (5) were evaluated using double-precision CPU calculations and their respective tolerances (Table 1; Notebook Table N1).

Automatic derivatives were compared with central finite differences for all 271 normalised parameter coordinates. The test used fixed initial states, a six-day synthetic forcing sequence and a prescribed target discharge Q_t^*. For this diagnostic only, the loss and derivative estimates were

$$L = \frac{1}{6}\sum_{t=1}^{6}(Q_t - Q_t^*)^2, \qquad A_j = \frac{\partial L}{\partial \tilde{\theta}_j}, \qquad F_j(h) = \frac{L(\tilde{\theta}_j + h) - L(\tilde{\theta}_j - h)}{2h}, \quad h = 10^{-6}. \tag{8}$$

The finite difference was also evaluated at $h/10$, yielding the noise estimate $a_j = |F_j(h) - F_j(h/10)|$. The original diagnostic used $|A_j - F_j|/\max(|A_j|, |F_j|, 10^{-10}) \leq 0.005$. A supplementary noise-aware criterion, $|A_j - F_j| \leq a_j + 0.005|F_j|$, was applied retrospectively and is reported separately in Notebook Table N2. Both diagnostics concern derivatives of the test loss with respect to normalised parameters; their acceptance rules and initially non-evaluable cases are specified in Sect. S2.2.

The six-day test was supplemented with five forcing and initial-state configurations that activate processes with weak responses under the initial conditions, while retaining the same loss and derivative comparison procedure. Fifteen unit-hydrograph routing parameters, evaluated outside the model-specific daily-step function, were additionally checked in the corresponding routing modules and in full simulations with routing enabled. One groundwater-exchange parameter was tested under a non-saturated initial state. The routing checks used longer sequences and corresponding prescribed discharge targets; all follow-up tests retained the same noise-aware comparison criterion (Sect. S2.2; Notebook Table N2).

Forward-output consistency was examined for each of the 36 models by comparing CPU execution with eager GPU and compiled GPU execution, giving 72 model–backend comparisons. Each comparison covered 531 catchments, 1000 days and double-precision calculations. The primary criterion was a maximum absolute discharge difference below 10^{-4} mm; the original protocol additionally allowed a maximum relative difference below 10^{-3}. A separate 366-day single-catchment hydrograph comparison complemented these tests (Fig. 3c; Notebook Table N1).

3.3 Computational benchmarks

Forward-execution performance was measured for all 36 models under a common workload of 531 catchments and 1000 days (Fig. 4). The comparison used one CPU thread and an NVIDIA RTX 3080 Ti GPU, with double-precision calculations. Each model–execution-mode combination was timed five times after an untimed run, synchronising the GPU for timing. The initial compilation cost was measured separately. Software and hardware configurations are documented in Sect. S4.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

For model m, the speed-up of compiled GPU relative to CPU and relative to eager GPU was calculated from the five-run median times:

$$s_m = \frac{\mathrm{median}_r(t_{m,r}^{\mathrm{CPU}})}{\mathrm{median}_r(t_{m,r}^{\mathrm{GPU,compiled}})}, \qquad g_m = \frac{\mathrm{median}_r(t_{m,r}^{\mathrm{GPU,eager}})}{\mathrm{median}_r(t_{m,r}^{\mathrm{GPU,compiled}})}. \tag{9}$$

The reported ensemble values are medians of the 36 model-specific ratios. For compilation overhead C_m, the number of repeated forward evaluations required to offset that cost was estimated as $N_m^* = C_m/(t_m^{\mathrm{CPU}} - t_m^{\mathrm{GPU,compiled}})$ (Fig. 4c; Notebook Table N5).

Scaling tests varied the number of concurrent catchments from 1 to 531 and simulation duration from 30 to 3650 days (Fig. 5). Forward throughput was defined as BT/t_{elapsed}, in catchment-days s^{-1}, where B and T denote batch size and number of simulated days. Additional experiments measured peak allocated GPU memory for forward simulation and for forward execution with gradient propagation. Reverse-mode cost was expressed as the ratio of combined forward–backward runtime to forward-only runtime. The supplementary resource tests and their hardware configurations are described in Notebook Table N6 and Sect. S4.

3.4 Independent calibration and differentiable parameter learning

In IC, each model was calibrated independently for each catchment using the covariance matrix adaptation evolution strategy (CMA-ES; Hansen and Ostermeier, 2001). Optimisation used the normalised parameter coordinates, active covariance adaptation and 10 independent starts per model–catchment pair, with a nominal budget of 300 generations. The first hydrological year (1 October 1980–30 September 1981) was repeated five times for a 1825-day warm-up, followed by calibration over the subsequent 14 hydrological years. Warm-up days were excluded from the objective. The retained parameter set maximised calibration-period KGE_ε across the starts. The calibration settings are summarised in Sect. S3.2.

In dPL, each model used a separate multilayer perceptron to map the 35 catchment attributes to its normalised parameter coordinates. The network comprised two hidden layers of 256 units with layer normalisation and GELU activations, followed by a sigmoid output and the parameter transform in Eq. (2). We trained each network across the 531 catchments using AdamW (Loshchilov and Hutter, 2019) and the mean catchment loss defined in Sect. 3.1. Training evaluated 365-day scored sequences selected within the pre-test period, each preceded by hydrological state warm-up. The main warm-up configuration used 730 days. Warm-up calculations were excluded from gradient propagation, and the resulting states were detached before the scored sequence. Training ran for up to 100 epochs, with early stopping permitted from epoch 50 onward. The retained network was selected by the lowest training-period objective. Learning rates, clipping, stopping criteria and related settings are specified in Sect. S3.3.

Both estimation routes used the same model-specific hydrological calculations and parameter bounds. IC permitted independent adjustment of basin parameters, whereas dPL constrained them through a shared, model-specific attribute-to-parameter mapping. Their optimisation procedures, parameter transformations and training warm-up protocols differed; the test period and

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.



performance definition were aligned. The corresponding model-level evaluation results and detailed protocol comparison are reported in Notebook Table N4. Algorithms A1 and A2 summarise the corresponding estimation workflows.

4 Results

4.1 Numerical reliability and differentiability across 36 models

Numerical verification covered water accounting, automatic derivatives and agreement between CPU and GPU forward calculations for all 36 model structures (Fig. 3; Table 1). All 188 model–forcing combinations satisfied the prescribed water-accounting tolerances. The largest accumulated residual was 2.43×10^{-11} mm, below the 10^{-8} mm reference threshold in Fig. 3a. The largest daily residual, 4.13×10^{-6} mm for HBV96, remained within that model's applicable tolerance of 5.5×10^{-6} mm.

Automatic derivatives were checked against central finite differences for the 271 normalised parameter coordinates. The initial six-day configurations did not activate every parameter, so verification was extended to additional forcing and initial-state conditions and to process-specific routing and groundwater-exchange tests. In the extended checks, 255 coordinates had nonzero derivatives that agreed with finite differences in the model's daily calculations; 15 routing coordinates were verified with active unit-hydrograph calculations, and one groundwater-exchange coordinate was verified under a non-saturated state. All 271 coordinates consequently met the adopted gradient-consistency criterion under at least one applicable condition (Fig. 3b; Table 1).

All 72 model–GPU-backend comparisons satisfied the prescribed absolute-or-relative discharge-agreement criterion; 71 met the strict absolute threshold, while the compiled-FlexIS case met the alternative relative criterion (Table 1). In a separate 366-day hydrograph comparison, the maximum absolute discharge difference between compiled GPU and CPU was 4.44×10^{-14} mm (Fig. 3c).

Table 1. Numerical verification of dMoT v1.0. The three test families use different units and acceptance criteria. Each parameter coordinate is counted once in the extended derivative result. The water-accounting tests use model- and duration-dependent tolerances for accumulated and daily residuals; the fixed 10^{-8} mm threshold in Fig. 3a applies to accumulated residuals.

Test family	Unit (n)	Pass	Fail	Main outcome
Water accounting	Model × forcing case (188)	188	0	Largest accumulated residual 2.43×10^{-11} mm; largest daily residual 4.13×10^{-6} mm (HBV96; tolerance 5.5×10^{-6} mm)
Automatic differentiation vs. finite differences	Model × parameter coordinate (271)	271	0	255 daily-step, 15 routing and 1 groundwater-exchange coordinates passed under applicable conditions
CPU–GPU forward agreement	Model × GPU backend (72)	72	0	71 below strict absolute threshold of 10^{-4} mm; the remaining compiled-FlexIS case met the original alternative relative-error threshold

Notes: Central finite differences used $h = 10^{-6}$. With $AD = \partial L/\partial\tilde{\theta}$ and $FD(h)$ the finite-difference estimate, the derivative criterion was $|AD - FD(h)| \leq |FD(h) - FD(h/10)| + 0.005|FD(h)|$; this noise-aware criterion was applied retrospectively. The compiled-FlexIS comparison reached 1.178×10^{-4} mm maximum absolute discrepancy but met the alternative relative threshold of 10^{-3} (observed maximum relative discrepancy 1.5×10^{-5}).

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

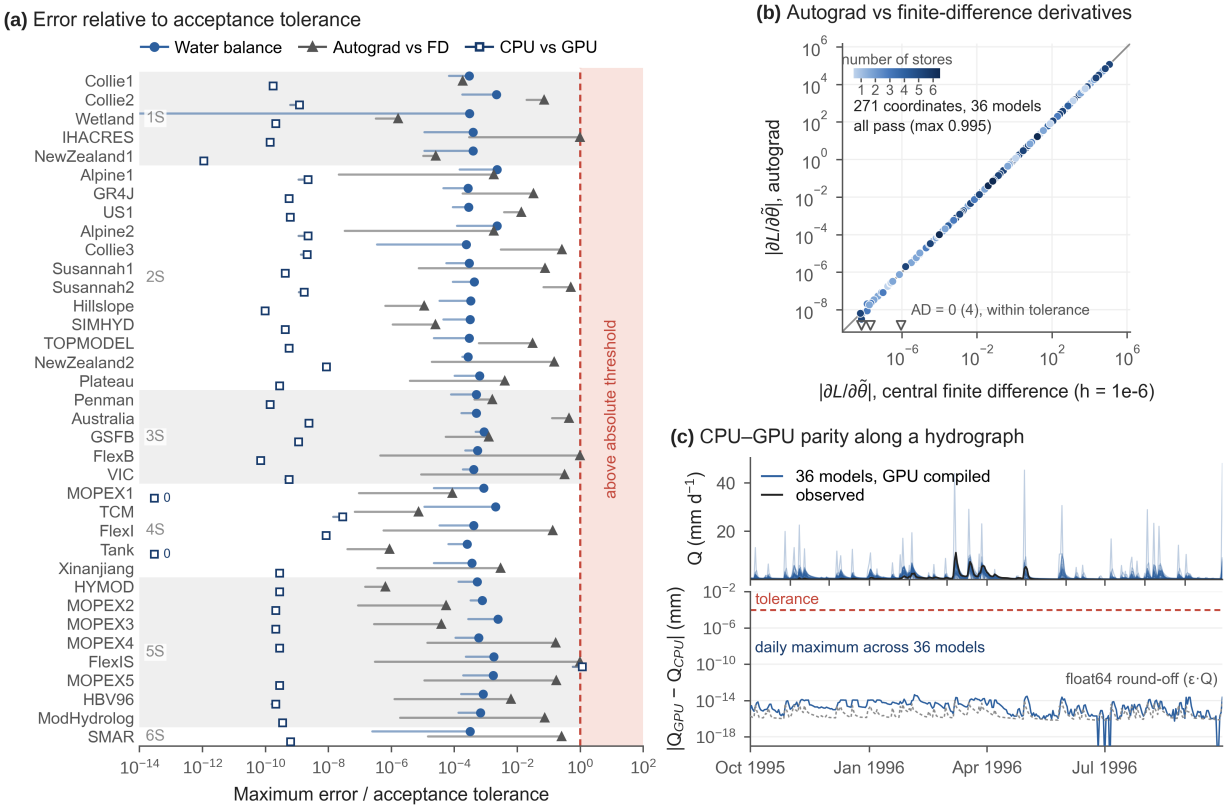


Figure 3. Numerical verification across the 36 models. (a) Model-specific water-accounting residuals, automatic-differentiation versus finite-difference discrepancies, and CPU–GPU forward-output differences, each shown relative to its displayed diagnostic tolerance. Exact-zero CPU–GPU differences (MOPEX1 and Tank) have no position on the logarithmic axis; they are drawn as open squares at the left axis limit and labelled 0. The CPU–GPU symbols are normalised by the strict absolute-error threshold; the single compiled-FlexIS comparison above unity nevertheless satisfies the alternative relative-error criterion. (b) Automatic-differentiation derivatives compared with central finite differences ($h = 10^{-6}$), with colour indicating the number of hydrological stores. Four coordinates (the fast–slow split parameters of FlexB, FlexI and FlexIS and the IHACRES α) have an identically zero autograd derivative in the plotted core-step scenario and are drawn as open triangles at the lower axis limit; their finite differences are at the numerical-noise floor, so they satisfy the same noise-aware criterion and are not verification failures. All 271 coordinates pass (maximum combined error ratio 0.995). The complete verification includes the additional applicable conditions summarised in Table 1. (c) Discharge simulations from 36 models and observed discharge (upper panel), with daily maximum CPU–GPU discharge differences across models (lower panel) for the separate hydrograph comparison.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

4.2 Computational acceleration through GPU compilation

For 531 catchments and 1000 simulated days, the median of the model-specific five-repeat median forward times was 380.9 ms on a single CPU thread, 770.2 ms on eager GPU and 123.1 ms on compiled GPU (Figs. 1c and 4). Executing the daily operations on GPU without compilation was therefore slower than the single-threaded CPU baseline for this workload, whereas compilation substantially reduced forward runtime.

The median model-specific compiled-GPU speed-up was 2.89-fold relative to CPU (range 1.11–8.10) and 5.90-fold relative to eager GPU (range 2.38–17.35; Fig. 4a). SMAR achieved the largest speed-up in both comparisons, at 8.10-fold and 17.35-fold, respectively. The benefits differed considerably among structures, with no monotonic relation to their numbers of hydrological stores (Fig. 4b).

Compilation incurred a one-time overhead, with a median of 3.00 s across models (range 1.43–6.75 s). When this overhead was included, compiled execution reached a median break-even point after 13.67 repeated forward runs (model-specific range 6.04–128.98; Fig. 4c). These estimates apply to repeated forward evaluations and exclude end-to-end calibration and differentiable-network training overhead.

4.3 Computational scaling and memory requirements

Compiled-GPU throughput increased nearly proportionally with the number of catchments in a batch (Fig. 5a). For 1000 simulated days, the median model-specific throughput increased from 8222 catchment-days s^{-1} for a single catchment to 4.31×10^6 catchment-days s^{-1} for 531 catchments, a 524.5-fold increase. Over the same range, median forward wall time changed only from 121.6 to 123.1 ms.

Simulation length introduced an approximately proportional increase in execution time (Fig. 5b). With 64 catchments, the median compiled-GPU runtime rose from approximately 5.1 ms at 30 days to 538.3 ms at 3650 days. At 3650 days, the corresponding median for 531 catchments was 539.1 ms.

Memory requirements increased when the computation graph was retained for automatic differentiation (Fig. 5c). Among the measured configurations, the highest PyTorch peak allocated memory was 359.3 MiB for SMAR with 531 catchments and 730 simulated days in a forward-plus-backward calculation. An approximate 480-byte-per-catchment-day reference line describes the growth of graph-retention memory within the measured range. These are peak PyTorch allocated-memory measurements, rather than total memory requirements of full dPL training. For 64 catchments and 90 days, the median forward-plus-backward runtime was 4.59 times the forward-only runtime (range 3.06–5.42), with the backward pass alone taking a median of 3.59 forward runtimes (Fig. 5d).

4.4 Large-sample calibration and differentiable parameter learning

All 36 models were applied under basin-wise independent calibration (IC) and differentiable parameter learning (dPL) to 531 CAMELS-US catchments, providing 19,116 paired model–catchment test evaluations (Fig. 6). Evaluation used later discharge records from the same catchments that contributed to dPL training. Based on the numerically stabilised Kling–Gupta efficiency,

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

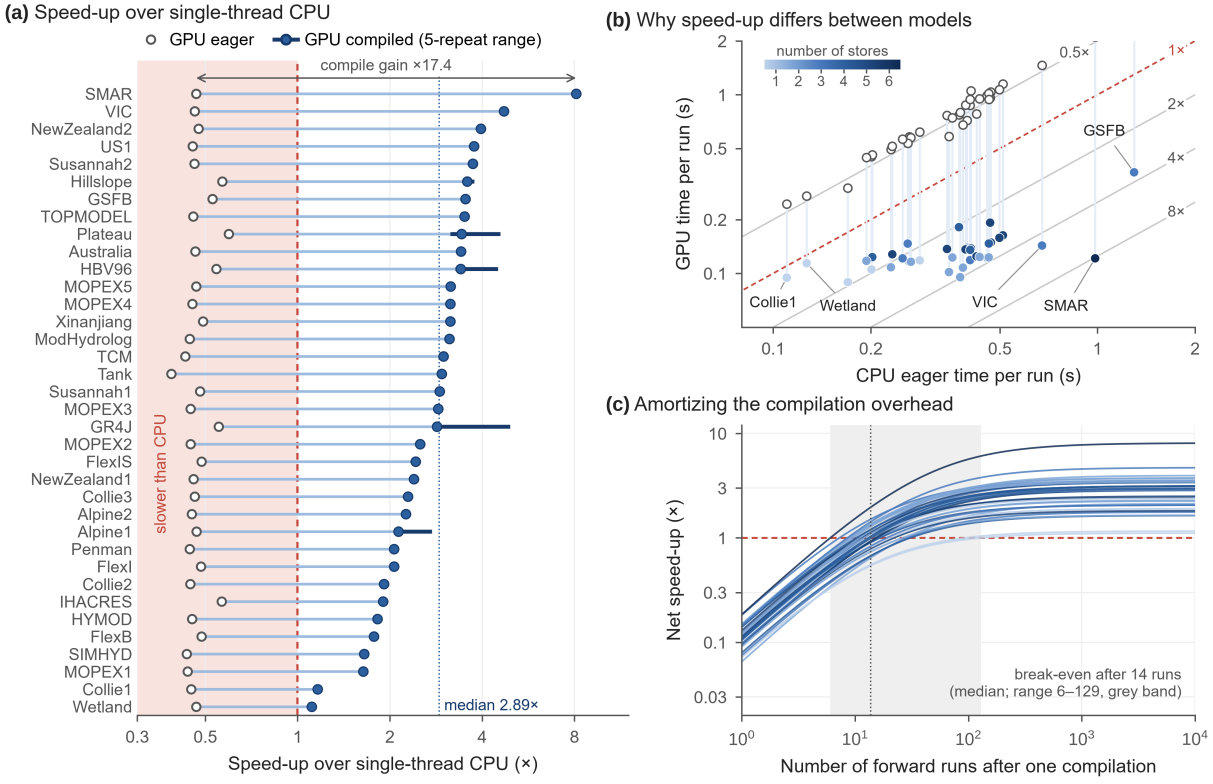


Figure 4. Forward-simulation acceleration and compilation overhead across the 36 models. All runtimes are for 531 catchments and 1000 daily time steps. (a) Eager-GPU (open circles) and compiled-GPU (filled circles) speed-ups relative to single-threaded CPU execution; short horizontal intervals indicate the five-repeat timing range for the compiled mode, and the vertical reference lines mark unity and the median compiled-GPU speed-up. (b) Eager- and compiled-GPU runtimes against CPU runtimes for each model, coloured by number of hydrological stores. Diagonal reference lines indicate fixed ratios of CPU to GPU runtime. (c) Net speed-up of compiled GPU as the one-time compilation overhead is distributed across repeated forward evaluations; the dotted line marks the median break-even count, and the shaded band spans model-specific break-even values.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

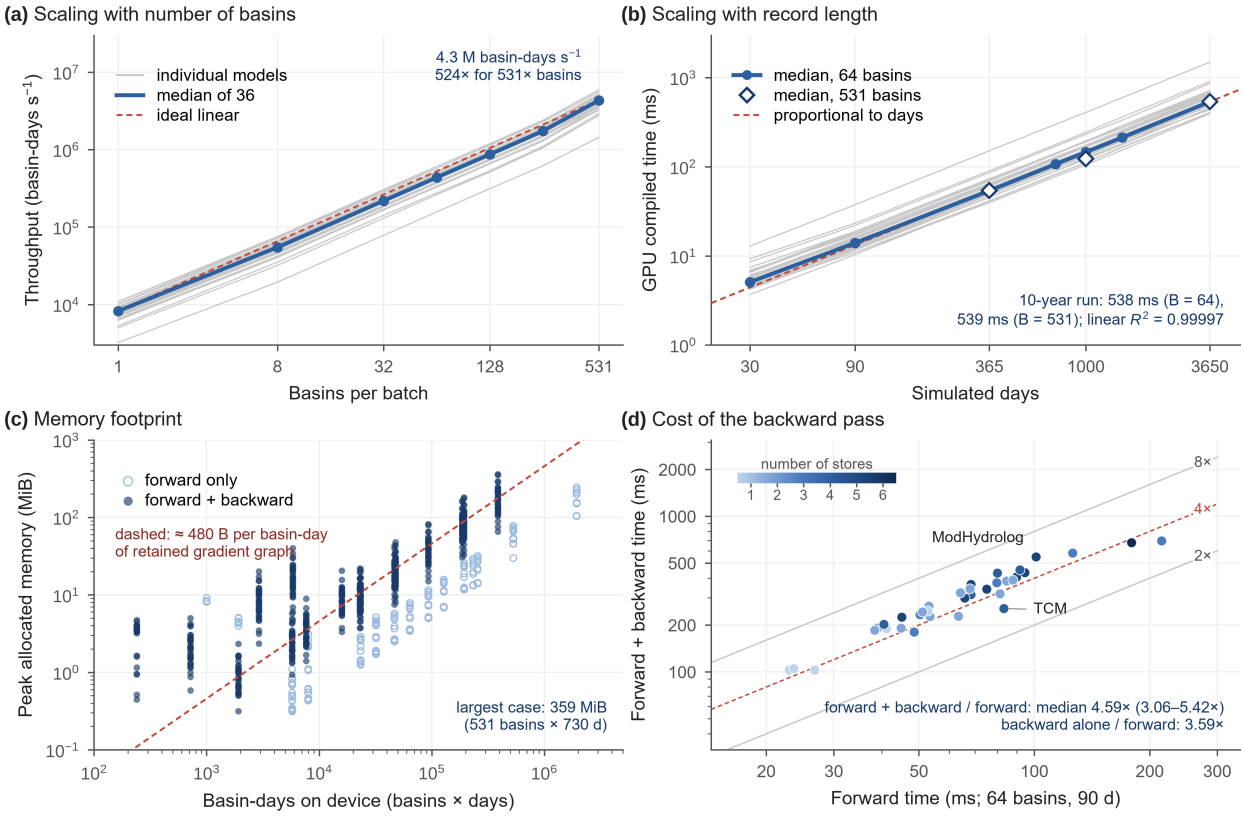


Figure 5. Scaling of compiled-GPU simulation and differentiation. (a) Forward throughput as catchment batch size increases for 1000-day simulations. Thin lines show individual models, the thick line their median and the dashed line ideal linear throughput scaling. (b) Compiled forward runtime across simulation lengths, showing model-level times and medians for 64 catchments; diamonds indicate the 531-catchment medians at selected lengths. The dashed line represents proportional scaling with simulation length. (c) Peak PyTorch allocated GPU memory against the number of catchment-days, shown separately for forward-only and forward-plus-backward execution. The dashed line is an approximate 480-byte-per-catchment-day guide for retained gradient graphs. Measurements were collected in two GPU sessions; retained first-call allocations can elevate some of the smaller-workload peaks. The plotted allocations do not represent full dPL training memory requirements. (d) Forward-plus-backward runtime against forward-only runtime for 64 catchments and 90 days, with points coloured by hydrological store count and diagonal lines indicating fixed time ratios.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

KGE_ε, the median of the 36 model-specific median test scores was 0.624 for IC and 0.612 for dPL (Fig. 6a). Comparing the two model-level medians separately, dPL matched or exceeded IC for 13 of 36 models.

Within the same model–catchment pairs, the median difference in KGE_ε (dPL minus IC) was -0.01265 across the 19,116 evaluations. Model-specific medians of paired differences ranged from -0.073 to $+0.010$, and four models—HBV96, NewZealand2, TOPMODEL and VIC—had nonnegative paired medians (Fig. 6a). The counts of 13 and four describe different summaries: a difference between two marginal medians and the median of paired differences, respectively. Spatial variation was also evident (Fig. 6b): 59.1% of catchments had an across-model median paired difference of at least -0.02, and 65.5% lay within ± 0.05.

For context, Fig. 6c compares the model-level results against those published for MARRMoT v1.0 by Knoben et al. (2020) over a maximum of 483 overlapping catchments (480–483 usable for individual models). The median numerical differences between study-specific model-level scores were $+0.050$ for IC and $+0.040$ for dPL, with Spearman model-rank correlations of 0.694 and 0.730, respectively. The two studies differ in their evaluation periods, calibration procedures, numerical implementations and KGE definitions. The comparison provides historical context for model-level performance patterns; its score differences cannot be attributed to the numerical implementation alone.

5 Discussion

5.1 Unifying structural diversity with differentiable modelling

dMoT v1.0 brings 36 established conceptual hydrological models into a common differentiable computing environment. The models retain different representations of water storage, runoff generation, and routing while sharing interfaces for meteorological forcing, parameter assignment, and simulation output (Figs. 1 and 2). Basin-parallel simulation and gradient-based parameter learning can therefore use the same hydrological calculations. A model structure needs to be implemented only once to support different parameter-estimation workflows, reducing differences introduced by maintaining separate simulation code for each task.

Frameworks such as FUSE and SUMMA established approaches for comparing alternative hydrological process formulations (Clark et al., 2008, 2015a, b), while MARRMoT assembled established rainfall–runoff models in a documented library (Knoben et al., 2019; Trotter et al., 2022). Multi-model studies have shown that different process representations can produce comparable streamflow performance (Knoben et al., 2020). In parallel, differentiable parameter learning has made it possible to estimate model parameters through trainable mappings that share information across catchments (Tsai et al., 2021; Feng et al., 2022). dMoT connects these developments by bringing a diverse collection of established conceptual structures into a computing environment that supports both conventional calibration and gradient-based learning.

The resulting scientific opportunity is to examine model structure and parameter-estimation strategy as separate choices in experimental design. Under a specified structure, researchers can compare alternative ways of estimating its parameters; under a specified estimation approach, they can investigate differences among hypotheses about storage, runoff generation, and routing. Such experiments can test whether patterns in learned parameters persist across structures or depend on the process formulations

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

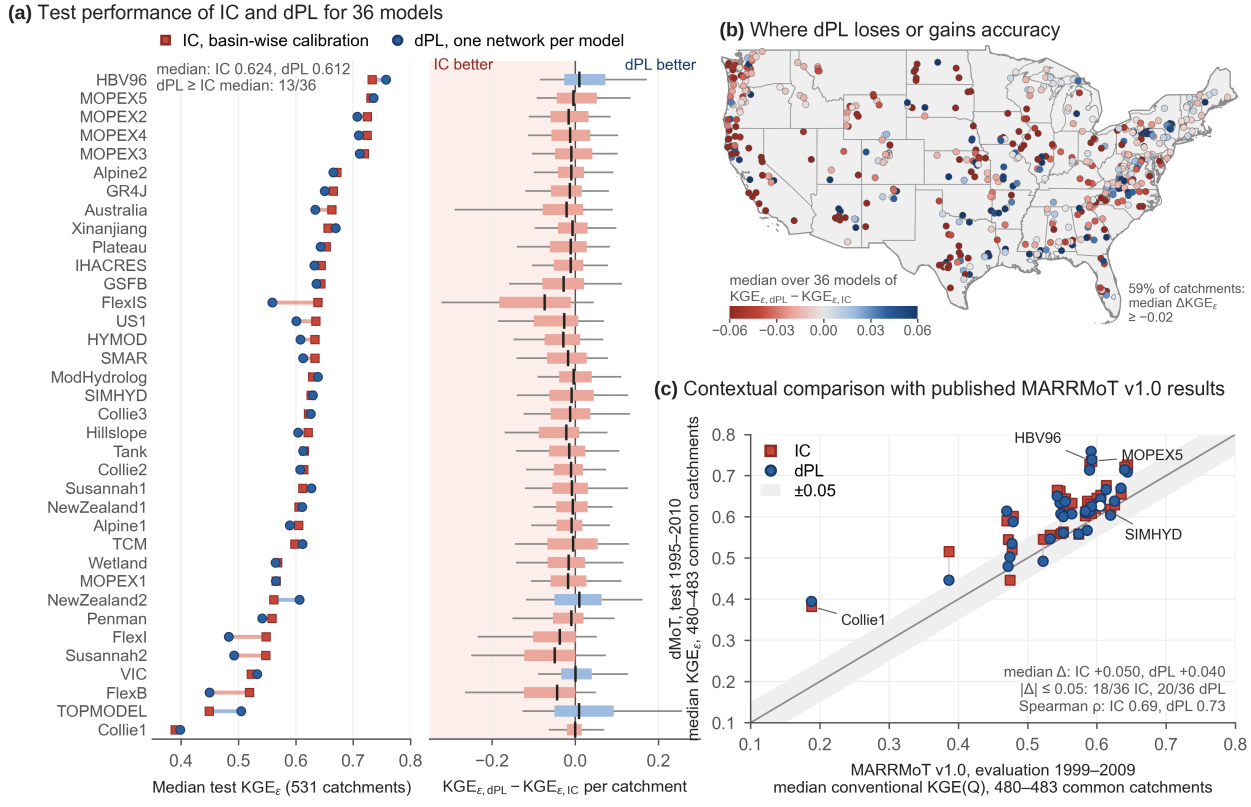


Figure 6. Large-sample demonstration of independent calibration and differentiable parameter learning. (a) Left: median test KGE_ε for each of the 36 models under IC (squares) and dPL (circles), calculated across 531 catchments. Right: within-catchment paired differences, $\Delta\mathrm{KGE}_\varepsilon = \mathrm{KGE}_{\varepsilon,\mathrm{dPL}} - \mathrm{KGE}_{\varepsilon,\mathrm{IC}}$, summarised by model; negative values favour IC and positive values favour dPL. (b) Spatial distribution of each catchment's median $\Delta\mathrm{KGE}_\varepsilon$ across the 36 models; the diverging colour scale is centred on zero. (c) Contextual comparison of model-level median scores with published MARRMoT v1.0 results (Knoben et al., 2020) for up to 483 overlapping catchments. The 1:1 line and grey ± 0.05 band are visual references only: the studies used different evaluation periods, calibration protocols and KGE definitions.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

used to generate them. The value of a common differentiable library therefore extends beyond executing more models: it makes structure-aware investigations of parameter learning feasible across large catchment samples.

5.2 Numerical reliability of differentiable conceptual models

Numerical calculations affect both discharge simulations and parameter inference. In dMoT, water transfers follow model-specific daily state updates and flux formulations. The synthetic water-accounting experiments show that the implemented daily updates preserve agreement between changes in storage and the specified water fluxes within the test tolerances (Fig. 3; Table 1). Applying the same accounting principle across the model catalogue provides a physically interpretable check on each structure's numerical implementation.

Differentiable learning additionally depends on the sensitivities propagated through those calculations. Song et al. (2024) showed how numerical formulations and gradient computation are connected in differentiable hydrological modelling, using an implicit, adjoint-based approach. In dMoT, automatic differentiation follows the implemented daily state recursion, and finite-difference comparisons independently assess derivatives of the test objective with respect to model parameters (Fig. 3; Table 1). Together, core-step, process-activating, and routing-aware tests covered all 271 parameter coordinates under the applicable verification conditions. This verification addresses a question that forward discharge checks alone cannot answer: whether the derivatives used during optimisation are consistent with the numerical model through which they are calculated. Evaluating these sensitivities model by model is particularly relevant when a common learning method is applied across contrasting process structures.

5.3 Large-sample computation and parameter-estimation workflows

The large-sample capacity of dMoT comes from representing catchments along a shared tensor dimension rather than evaluating them through separate model implementations. Each catchment retains its own forcing, parameters, and hydrological states, while the same sequence of process calculations advances the entire batch. In the compiled-GPU benchmark, increasing the batch from one to 531 catchments raised forward throughput almost in proportion to batch size, with little change in elapsed time for the tested 1000-day simulation (Fig. 5a). This reduces the marginal cost of including additional catchments in a batched forward calculation and supports comparisons across large samples under consistent model definitions.

Repeated model evaluation creates a different computational requirement. Independent calibration (IC) repeatedly evaluates a hydrological structure with different candidate parameters. The PyTorch compilation mechanism optimises recurring computations (Ansel et al., 2024); in dMoT, compiling the model-specific daily step produced a median 2.89-fold forward speed-up relative to single-threaded CPU execution for the 531-catchment, 1000-day workload (Fig. 4). At the model-level median, the initial compilation cost was recovered after approximately 14 identical forward evaluations. The relevance to parameter search lies in this repeated use of the same hydrological calculations, where compilation costs can be spread over successive evaluations.

Differentiable parameter learning (dPL) makes different demands on the same execution environment. It trains a shared catchment-attribute-to-parameter mapping by propagating gradients through the hydrological state calculations (Tsai et al.,

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

2021; Feng et al., 2022). The resulting forward–backward computation requires additional execution time and storage for the differentiation graph, as quantified by the memory and reverse-pass benchmarks (Fig. 5c, d). The application of both estimation workflows across the 36-model catalogue (Fig. 6) demonstrates that dMoT can accommodate repeated forward optimisation and gradient-based regional parameter learning through the same set of hydrological implementations. Their different workloads make batch size, compilation reuse, and the length of the differentiated sequence practical considerations in large-sample experiments.

5.4 Scope, limitations, and future directions

The catchment experiment defines the application-level scope of the present release. Across the 36 models, the median of the model-level median test scores for the stabilised Kling–Gupta efficiency was 0.624 under IC and 0.612 under dPL (Fig. 6). These outcomes were obtained with different optimisation procedures, training warm-up periods, and parameter transformations, so they describe the performance attained under two specified workflows rather than a controlled ranking of estimation strategies. The median paired dPL–IC difference in $\mathrm{KGE}_{\varepsilon}$ across the 19,116 model–catchment evaluations was -0.01265. The evaluation period was held out in time, but all 531 catchments had contributed to dPL training. Consequently, the results characterise later-period performance at previously represented catchments rather than spatial transfer to ungauged basins.

Numerical verification applies to the discrete implementations documented for dMoT. The library uses independent PyTorch calculations of established conceptual structures, with model-specific and numerical differences from corresponding MARRMoT implementations recorded in the Appendix (Tables A1 and A2). A matched-forcing, fixed-parameter comparison of daily trajectories across the two libraries has not yet been conducted. The historical MARRMoT comparison (Fig. 6c) therefore provides context for broad model-performance patterns rather than a direct test of numerical equivalence (Knoben et al., 2020).

The computational benchmarks similarly describe measured workloads and hardware configurations. The principal forward comparison used double-precision calculations on a single CPU thread and an RTX 3080 Ti GPU (Fig. 4), while the scaling and differentiation tests covered specified batch sizes and simulation periods (Fig. 5). Reported forward-execution speed-ups do not measure the elapsed time of a complete IC search or dPL training run. Compilation overhead and retained-graph memory also make the preferred execution mode dependent on the workload. Additional hardware comparisons and end-to-end profiling would extend the practical guidance available for these settings.

Future work can extend the current verification and application programme in three directions. Spatially held-out catchment experiments, supported where possible by independent observations of internal hydrological states, would test parameter transfer and strengthen process interpretation. Matched-input, fixed-parameter comparisons with reference implementations would establish a more direct account of numerical correspondence. Finally, broader training benchmarks and memory-management strategies could support longer differentiation horizons and larger catchment collections. These extensions would build on dMoT's central capability: representing alternative conceptual hydrological structures within one documented, differentiable environment for large-sample modelling.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

6 Conclusions

We developed dMoT v1.0, a differentiable, basin-parallel PyTorch library implementing 36 established conceptual rainfall–runoff models. The library preserves model-specific representations of water storage, runoff generation, and routing while providing common interfaces for simulation and parameter estimation. Its main contribution is to make a diverse collection of hydrological process representations available for both independent catchment calibration (IC) and differentiable parameter learning (dPL) through the same model implementations. This enables model structure and parameter-estimation strategy to be varied as separate choices in large-sample hydrological experiments.

Numerical verification confirmed water-budget consistency and agreement between automatic-differentiation and finite-difference gradients under the tested conditions. Computational benchmarks demonstrated a median 2.89-fold forward-execution speed-up with compiled GPU execution relative to single-threaded CPU execution, alongside effective basin batching and the additional computational requirements of gradient-based learning. The application across 531 CAMELS-US catchments demonstrated the use of both IC and dPL with all 36 model structures. By combining structural diversity, differentiable computation, and basin-parallel execution, dMoT provides a common computational basis for systematically investigating how hydrological model assumptions and parameter-estimation strategies influence modelling outcomes.

Code and data availability. The dMoTpy v1.0.1 software and the complete reproduction archive for this study are available from Zenodo at https://doi.org/10.5281/zenodo.24110890 (Jing et al., 2026). The software is released under GPL-3.0-or-later, compatible with the GPL-3.0 licence of the MARRMoT source material. The archive includes the corresponding model inputs and reproduction materials, together with a README and Technical Notebook describing the software environment and step-by-step reproduction procedure. Other versions of the source code are available at https://github.com/chooron/dmot-research. Meteorological forcing and streamflow observations were obtained from the CAMELS dataset (Newman et al., 2015; Addor et al., 2017), using the CAMELS extraction distributed by the MHPI group (https://mhpi-spatial.s3.us-east-2.amazonaws.com/mhpi-release/data/1-camels.zip). Static catchment attributes used for dPL were compiled from CAMELS (Addor et al., 2017) and Caravan (Kratzert et al., 2023).

Appendix A: Model inventory, implementation differences and estimation workflows

A1 Model inventory and implementation differences

The 36 hydrological models share an interface for daily simulation, but differ in state variables, parameter dimensions, process combinations and routing configurations. Table A1 identifies the model-specific configurations used in this study. It counts hydrological stores separately from unit-hydrograph memory and lists the process categories shown in Fig. 2. Detailed process equations and parameter bounds are documented in Sects. N2–N4 of the companion Technical Notebook.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Table A1: Inventory and structural characteristics of the 36 conceptual hydrological models implemented in dMoT v1.0. Models are listed in the order of Fig. 2 (by store count, then parameter count). "Stores" counts the state variables advanced by the daily recursion; "Par." counts the free parameter coordinates, which sum to the 271 used throughout the paper. "Routing" gives the configured placement of unit-hydrograph (UH) routing and the largest lag admitted by the parameter bounds of each model; "routing store" denotes in-model routing storage without a separate unit hydrograph. Main processes are the process categories of Fig. 2b.

ID	Model	Par.	Stores	Routing	Main processes
1	Collie1	1	1	–	soil moisture
2	Collie2	4	1	–	soil moisture, interflow
3	Wetland	4	1	–	interception, soil moisture, baseflow
4	IHACRES	6	1	outlet UH, 5 / 30 d	soil moisture
5	NewZealand1	6	1	–	soil moisture, interflow, baseflow
6	Alpine1	4	2	–	snow, soil moisture, baseflow
7	GR4J	4	2	internal UH, $x_4 \leq 15$ d, dual branch	soil moisture
8	US1	5	2	–	interception, soil moisture, baseflow
9	Alpine2	6	2	–	snow, soil moisture, interflow, baseflow
10	Collie3	6	2	–	soil moisture, interflow, baseflow
11	Susannah1	6	2	–	soil moisture, interflow, baseflow
12	Susannah2	6	2	–	soil moisture, interflow
13	Hillslope	7	2	outlet UH, 120 d	interception, soil moisture, baseflow
14	SIMHYD	7	2	–	interception, soil moisture, interflow, baseflow
15	TOPMODEL	7	2	–	soil moisture, interflow, baseflow
16	NewZealand2	8	2	outlet UH, 30 d	interception, soil moisture, interflow, baseflow
17	Plateau	8	2	outlet UH, 120 d	interception, soil moisture, baseflow
18	Penman	4	3	routing store	soil moisture, baseflow
19	Australia	8	3	–	soil moisture, interflow, baseflow
20	GSFB	8	3	–	soil moisture, interflow, baseflow
21	FlexB	9	3	internal UH, 5 / 15 d	soil moisture, baseflow
22	VIC	10	3	–	interception, soil moisture, baseflow
23	MOPEX1	5	4	routing store	soil moisture, baseflow
24	TCM	6	4	routing store	soil moisture, baseflow
25	FlexI	10	4	internal UH, 5 / 15 d	interception, soil moisture, baseflow
26	Tank	12	4	–	soil moisture, interflow, baseflow
27	Xinanjiang	12	4	routing store	soil moisture, interflow, baseflow
28	HYMOD	5	5	routing store	soil moisture, baseflow
29	MOPEX2	7	5	routing store	snow, soil moisture, baseflow
30	MOPEX3	8	5	routing store	snow, soil moisture, baseflow

continued on next page

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

ID	Model	Par.	Stores	Routing	Main processes
31	MOPEX4	10	5	routing store	snow, interception, soil moisture, baseflow
32	FlexIS	12	5	internal UH, 5 / 15 d	snow, interception, soil moisture, baseflow
33	MOPEX5	12	5	routing store	snow, interception, soil moisture, baseflow
34	HBV96	15	5	outlet UH, 120 d	snow, soil moisture, interflow, baseflow
35	ModHydrolog	15	5	–	interception, soil moisture, interflow, baseflow
36	SMAR	8	6	outlet UH, 120 d gamma	soil moisture, baseflow

Notes: Store counts exclude memory associated with separately configured unit-hydrograph convolution; routing stores explicitly represented among the hydrological states remain included in "Stores" (Penman, MOPEX1–5, TCM, HYMOD, Xinanjiang). For HBV96, the five stores represent the snowpack, snow liquid water, soil, upper zone and lower zone; its core daily function outputs $q_0 + q_1$ before outlet-UH routing, and the `maxbas` parameter does not enter that function's state updates. The UH classification refers to the registered external routing component, not to an additional hydrological store. A dash denotes no separately classified UH pathway or routing store. Model-specific equations and parameter bounds are documented separately.

The implementation is based on MARRMoT model families rather than a step-by-step numerical reproduction of the MATLAB implementations. In particular, the ordered daily state updates can evaluate later fluxes using states modified earlier within the same day (Eq. (3)). Table A2 records selected verified structural, process and numerical differences without treating unlisted models as numerically identical.

Table A2: Principal structural and numerical differences between dMoT v1.0 and the corresponding MARRMoT implementations. "Reference" describes the MARRMoT formulation introduced in Sect. 2; "dMoT" describes the implementation in this paper. The type column separates structural differences (store number, parameter set), process-formulation differences, and numerical protection. Only source-supported differences are listed; absence from this table does not establish equivalence to MARRMoT. These comparisons do not establish day-by-day numerical equivalence with the MATLAB implementation for identical parameters and forcing.

Model	Reference formulation	dMoT implementation	Type	Implication
All	The model functions return store derivatives and process fluxes evaluated at the state supplied by the numerical solver; some equations additionally use saved previous-step values (e.g. HBV96's $S_{2,\mathrm{old}}$).	Each daily step evaluates fluxes in a fixed documented order and updates stores sequentially, so later fluxes of the same day see updated stores (Eq. 3).	Integration scheme	A daily increment is an operator split rather than the reference solution of the store ODEs; identical parameters and forcing therefore need not reproduce the reference discharge step by step.
SIMHYD	Three stores (interception, soil moisture, groundwater) and ten fluxes.	Two stores (soil moisture, groundwater); the interception capacity acts inside the daily step instead of a separate store.	Structure	The number and interpretation of storage states differ; direct state-level comparisons are not one-to-one.

continued on next page

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Model	Reference formulation	dMoT implementation	Type	Implication
IHACRES	Seven parameters, including a pure delay of total flow (τ_d, up to 119 d) applied by a delay unit hydrograph, with fast and slow delays bounded by 700 d.	Six parameters without the total-flow delay; the fast and slow delays are exponential unit hydrographs bounded by 5 and 30 d.	Parameter set	The two libraries do not share a parameter space, and the routing response is not comparable one to one.
HBV96	The five-store ODE formulation calculates snowpack liquid release through `infiltration_3` using $P_{\mathrm{rain}} + M$ and the current S_2, and `excess_1` using $S_{2,\mathrm{old}}$ and $\mathrm{WHC}\,S_1$; recharge uses their combined release and the solver state S_3.	After updating the snowpack, the daily routine releases liquid above $\mathrm{WHC}\,S_1^{\mathrm{new}}$ from a provisional S_2. Capillary transfer, evapotranspiration, recharge, percolation and both runoff fluxes then use sequentially updated states with explicit availability caps.	Equation, sequence	The snow-release and downstream fluxes are evaluated against different within-day states. The dMoT release term aggregates liquid release that is represented by two separate MARRMoT fluxes; numerical equivalence is not assumed.
Penman	Lower-zone evaporation scales the full potential rate by a below-threshold gate applied to the upper store at the start of the step; recharge overflow is tested with a smooth saturation function.	The gate is applied to the upper store after its own update and scales the potential rate remaining after upper-store evaporation; overflow uses a hard positive-part test.	Equation, sequence	Both the gate argument and the available potential rate differ, so lower-zone evaporation is not numerically equivalent.
TCM	Deep evaporation uses the same start-of-step below-threshold gate on the upper store, and the slow outflow is capped by the available storage.	The gate uses the updated upper store. The slow-outflow flux lacks the reference's explicit storage-availability cap; a nonnegative floor is applied to the updated store.	Equation, sequence	The evaporation timing and the explicit storage-availability restriction differ; their quantitative effect on a matched run is not established by this comparison.
GSFB	Recharge and the subsequent runoff and drainage fluxes follow the reference call sequence without availability limits beyond those in the reference formulae.	Recharge, saturation excess, evaporation, infiltration, baseflow and percolation are each limited by the water available at that point in the sequence, and the shared recharge threshold floor is bypassed locally.	Numerical protection	Availability caps change the near-empty-store flux calculation; the actual magnitude and sign of discharge differences are not inferred here.
Plateau	Saturation excess is evaluated from the start-of-step unsaturated storage.	Evaluated from the post-inflow, post-evaporation storage and additionally capped by the remaining storage water and by the current inflow.	Equation, protection	Runoff depends on an intermediate within-day state and additional availability limits; its net difference from the reference is not quantified here.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Notes: Model names follow Fig. 2. For HBV96, the snowfall/rainfall ramp is not identified as a distinct equation solely because the PyTorch implementation derives rainfall as P minus snowfall; exact helper-function and threshold-boundary equivalence is not established by this comparison. "Equation" denotes a change to a process formulation, "sequence" a change in where within the daily order a quantity is evaluated, "structure" a change in the number of stores, "parameter set" a change in the free parameters, "integration scheme" a change in the time integration of the storage equations, and "numerical protection" a bound that limits a flux to the water actually available. This table is a selective comparison rather than a full model-equation inventory. The reference implementations are the vendored MARRMoT MATLAB sources recorded in the accompanying provenance documentation.

A2 Parameter-estimation workflows

Algorithms A1 and A2 summarise the two parameter-estimation procedures applied to the common hydrological simulator. Independent calibration (IC) searches for a parameter vector separately for each model–catchment pair; differentiable parameter learning (dPL) generates catchment-specific parameters from a neural mapping shared across the catchments for each model. The algorithms describe the computational sequence; the calibration periods and warm-up protocols are described in Sect. 3.4 and Sect. S3 of the Supplement.

Algorithm A1 Workflow of basin-wise independent calibration (IC).

Require: Hydrological model m; catchment forcing $\boldsymbol{x}_t$ and observed discharge O; model-specific parameter bounds.

Ensure: Independently calibrated parameters $\boldsymbol{\theta}_{m,b}$ for each catchment b.

for all catchments b **do**
  Initialise the best-scoring parameter set for catchment b.
  for all independent CMA-ES starts **do**
    Initialise CMA-ES over normalised coordinates $\tilde{\boldsymbol{\theta}}_{m,b}$.
    while the optimisation stopping criterion is not met **do**
      Propose candidate coordinates and map them linearly to physical parameters.
      Set the prescribed initial hydrological states and routing memory.
      Simulate five repetitions of the first hydrological year (1 October 1980–30 September 1981) using model-specific daily updates and routing (Eq. 3); this gives the 1825-day warm-up, which is excluded from the objective.
      Simulate 1 October 1981–30 September 1995 (14 hydrological years) with model-specific daily updates and routing (Eq. 3).
      Evaluate the scored discharge Q against O using $\mathrm{KGE}_{\varepsilon}$ (Eq. 7).
      Update CMA-ES and retain the best-scoring candidate.
    end while
  end for
  Set $\boldsymbol{\theta}_{m,b}$ to the best candidate across starts.
end for
return $\{\boldsymbol{\theta}_{m,b}\}_b$.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Algorithm A2 Workflow of differentiable parameter learning (dPL).

Require: Hydrological model m; static catchment attributes $\boldsymbol{X}_b$, forcing $\boldsymbol{x}_t$ and observed discharge O; model-specific parameter bounds.

Ensure: Shared network weights W_m and catchment parameters $\boldsymbol{\theta}_{m,b}$.

Initialise the model-specific attribute-to-parameter network g_{W_m} and its optimiser.
while the training stopping criterion is not met **do**
  Select the catchments and a 365-day scored sequence from the pre-test record for this training update.
  for all selected catchments b **do**
    $\boldsymbol{z}_{m,b} \leftarrow g_{W_m}(\boldsymbol{X}_b)$; $\tilde{\boldsymbol{\theta}}_{m,b} \leftarrow \sigma(\boldsymbol{z}_{m,b})$.
    $\boldsymbol{\theta}_{m,b} \leftarrow \boldsymbol{\phi}_m^{\mathrm{dPL}}(\tilde{\boldsymbol{\theta}}_{m,b})$ (Eq. 2).
    Warm up using the model-specific period (730 d in the main configuration; 365 d for Penman); detach terminal states and routing memory before differentiation.
    Simulate the 365-day scored sequence using model-specific daily updates (Eq. 3).
  end for
  Compute $\mathcal{L} \leftarrow 1 - \overline{\mathrm{KGE}_\varepsilon}$ over eligible catchments (Eq. 7).
  Backpropagate $\partial\mathcal{L}/\partial W_m$ through the scored simulation and mapping (Eq. 6).
  Update W_m using AdamW.
end while
Retain the network W_m selected by the training objective.
Generate $\{\boldsymbol{\theta}_{m,b}\}_b$ from the retained network and Eq. 2.
return W_m and $\{\boldsymbol{\theta}_{m,b}\}_b$.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Appendix B: Static catchment attributes

Table B1 lists the 35 static catchment attributes in their network-input order, including source fields, units and transformations preceding standardisation. Twenty-four variables retain the CAMELS-US attributes and the remaining 11 are replaced or recomputed using Caravan/HydroATLAS-derived information. The table records the actual preprocessing conventions of the analysed inputs, including the stored scale of terrain slope and the numeric treatment of class codes; it does not imply that these quantities were independently rescaled to physical units before training.

Table B1: Static catchment attributes used for differentiable parameter learning. The 35 attributes follow the fixed model-input order; units refer to stored numeric values before network-input standardisation. All inputs are subsequently z-score standardised; the preprocessing column lists additional prior transformations.

No.	Attribute	Description	Unit	Source field	Extra preprocessing
1	p_mean	Mean daily precipitation	mm d^{-1}	H: `pre_mm_syr`	annual /365
2	pet_mean	Mean daily potential evapotranspiration	mm d^{-1}	H: `pet_mm_syr`	annual /365
3	p_seasonality	Moisture-index seasonality	–	C: `seasonality`	–
4	frac_snow	Fraction of precipitation falling as snow	–	C: `frac_snow`	–
5	aridity	Ratio of mean PET to mean precipitation	–	Derived: PET/P	–
6	high_prec_freq	High-precipitation day frequency	days yr^{-1}	CAMELS-US	–
7	high_prec_dur	Mean duration of high-precipitation events	days	CAMELS-US	–
8	low_prec_freq	Dry-day frequency	days yr^{-1}	CAMELS-US	–
9	low_prec_dur	Mean duration of dry periods	days	CAMELS-US	–
10	elev_mean	Mean catchment elevation	m	H: `ele_mt_sav`	–
11	slope_mean	Mean catchment terrain slope	0.1°	H: `slp_dg_sav`	–
12	area_gages2	Catchment drainage area (GAGES-II)	km^2	CAMELS-US	shifted ln
13	frac_forest	Forest-covered catchment fraction	–	H: `for_pc_sse`	$\times 0.01$
14	lai_max	Maximum monthly mean leaf-area index	–	CAMELS-US	–
15	lai_diff	Seasonal range of monthly mean leaf-area index	–	CAMELS-US	–
16	gvf_max	Maximum monthly green-vegetation fraction	–	CAMELS-US	–
17	gvf_diff	Seasonal range of green-vegetation fraction	–	CAMELS-US	–
18	dom_land_cover_frac	Area fraction of dominant land cover	–	CAMELS-US	–
19	dom_land_cover	Dominant land-cover class	class code	CAMELS-US	–
20	root_depth_50	Depth containing 50% of root density	m	CAMELS-US	–
21	soil_depth_pelletier	Depth to bedrock (Pelletier)	m	CAMELS-US	–
22	soil_depth_statsgo	Soil depth (STATSGO)	m	CAMELS-US	–
23	soil_porosity	Volumetric soil porosity	–	CAMELS-US	–
24	soil_conductivity	Saturated soil hydraulic conductivity	cm h^{-1}	CAMELS-US	shifted ln
25	max_water_content	Maximum soil water content (depth equivalent)	m	CAMELS-US	–
26	sand_frac	Sand fraction in soil	%	H: `snd_pc_sav`	–
27	silt_frac	Silt fraction in soil	%	H: `slt_pc_sav`	–
28	clay_frac	Clay fraction in soil	%	H: `cly_pc_sav`	–

continued on next page

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

No.	Attribute	Description	Unit	Source field	Extra preprocessing
29	geol_1st_class	Dominant geological class	class code	CAMELS-US	–
30	glim_1st_class_frac	Fraction in dominant geological class	–	CAMELS-US	–
31	geol_2nd_class	Second-most-common geological class	class code	CAMELS-US	–
32	glim_2nd_class_frac	Fraction in second geological class	–	CAMELS-US	–
33	carbonate_rocks_frac	Carbonate sedimentary rock fraction	–	CAMELS-US	–
34	geol_porosity	Subsurface porosity	–	CAMELS-US	–
35	geol_permeability	$\log_{10}$ subsurface permeability	$\log_{10}(m^2)$	CAMELS-US	shifted ln

Notes: H denotes a HydroATLAS attribute column delivered via Caravan where a basin match exists (Linke et al., 2019); C denotes a Caravan climate statistic derived from ERA5-Land (Kratzert et al., 2023). The other 24 attributes are from CAMELS-US (Addor et al., 2017). The three skewed inputs (area, soil conductivity and geology permeability) use a data-dependent shifted natural logarithm before standardisation; the shift is not always one. The stored name `p_seasonality` denotes Caravan `seasonality`, a moisture-index statistic (range 0–2), rather than the CAMELS-US precipitation-timing index. HydroATLAS `slp_dg_sav` is stored as 10 times the slope in degrees and receives no further rescaling. HydroATLAS sand, silt and clay are retained as percentages; CAMELS-US `geol_permeability` is originally $\log_{10}$ permeability in m^2. Non-finite source values are replaced by zero, class labels are passed as numeric encodings, and all 35 inputs receive column-wise z-scoring.

Author contributions. Xin Jing: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft; Jungang Luo: Conceptualization, Supervision, Writing – review & editing; Mingxiang Yang: Methodology, Investigation, Writing – review & editing; Yaoxuan Xiao: Validation, Visualization, Writing – review & editing; Xue Yang: Methodology, Supervision, Writing – review & editing.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. We thank the developers of MARRMoT for making their conceptual hydrological model toolbox openly available, and Knoben et al. (2025) for the study that motivated this work. We acknowledge the earlier research on differentiable parameter learning (dPL) that informed our approach. We are grateful to Chaopeng Shen and the Multi-scale Hydrology, Processes and Intelligence (MHPI) group for sharing their differentiable HBV examples and the δMG (`generic_deltamodel`) framework, which supported the development of our training workflow. We also acknowledge the developers of EvoTorch for providing GPU-accelerated evolutionary optimization tools that facilitated efficient basin-wise calibration.

AI tools statement. Claude (Anthropic) and ChatGPT (OpenAI) were used to assist with language editing and improving manuscript clarity. Codex (OpenAI) was used to assist with the development of analysis and plotting scripts. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the manuscript and the reported results.

Financial support. This work was supported by the National Natural Science Foundation of China (grant no. 52209035); the Science and Technology Program of the Shaanxi Provincial Department of Water Resources (grant no. 2026SLKJ-4); the Shaanxi Provincial Science and Technology Plan Project (grant no. 2026SYS-SYSZD-73); and the Doctoral Dissertation Innovation Fund of Xi'an University of Technology (grant no. BC202610).

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

References

Addor, N. and Melsen, L. A.: Legacy, rather than adequacy, drives the selection of hydrological models, Water Resources Research, 55, 378–390, https://doi.org/10.1029/2018WR022958, 2019.

Addor, N., Newman, A. J., Mizukami, N., and Clark, M. P.: The CAMELS data set: catchment attributes and meteorology for large-sample studies, Hydrology and Earth System Sciences, 21, 5293–5313, https://doi.org/10.5194/hess-21-5293-2017, 2017.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., and Chintala, S.: PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation, in: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, pp. 929–947, ACM, https://doi.org/10.1145/3620665.3640366, 2024.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M.: Automatic differentiation in machine learning: a survey, Journal of Machine Learning Research, 18, 1–43, https://jmlr.org/papers/v18/17-468.html, 2018.

Beck, H. E., van Dijk, A. I. J. M., de Roo, A., Miralles, D. G., McVicar, T. R., Schellekens, J., and Bruijnzeel, L. A.: Global-scale regionalization of hydrologic model parameters, Water Resources Research, 52, 3599–3622, https://doi.org/10.1002/2015WR018247, 2016.

Beven, K.: A manifesto for the equifinality thesis, Journal of Hydrology, 320, 18–36, https://doi.org/10.1016/j.jhydrol.2005.07.007, 2006.

Clark, M. P., Slater, A. G., Rupp, D. E., Woods, R. A., Vrugt, J. A., Gupta, H. V., Wagener, T., and Hay, L. E.: Framework for Understanding Structural Errors (FUSE): A modular framework to diagnose differences between hydrological models, Water Resources Research, 44, 2007WR006 735, https://doi.org/10.1029/2007WR006735, 2008.

Clark, M. P., Kavetski, D., and Fenicia, F.: Pursuing the method of multiple working hypotheses for hydrological modeling, Water Resources Research, 47, 2010WR009 827, https://doi.org/10.1029/2010WR009827, 2011.

Clark, M. P., Nijssen, B., Lundquist, J. D., Kavetski, D., Rupp, D. E., Woods, R. A., Freer, J. E., Gutmann, E. D., Wood, A. W., Brekke, L. D., Arnold, J. R., Gochis, D. J., and Rasmussen, R. M.: A unified approach for process-based hydrologic modeling: 1. Modeling concept, Water Resources Research, 51, 2498–2514, https://doi.org/10.1002/2015WR017198, 2015a.

Clark, M. P., Nijssen, B., Lundquist, J. D., Kavetski, D., Rupp, D. E., Woods, R. A., Freer, J. E., Gutmann, E. D., Wood, A. W., Gochis, D. J., Rasmussen, R. M., Tarboton, D. G., Mahat, V., Flerchinger, G. N., and Marks, D. G.: A unified approach for process-based hydrologic modeling: 2. Model implementation and case studies, Water Resources Research, 51, 2515–2542, https://doi.org/10.1002/2015WR017200, 2015b.

Coxon, G., Freer, J., Lane, R., Dunne, T., Knoben, W. J. M., Howden, N. J. K., Quinn, N., Wagener, T., and Woods, R.: DECIPHeR v1: Dynamic fluxEs and ConnectIvity for Predictions of HydRology, Geoscientific Model Development, 12, 2285–2306, https://doi.org/10.5194/gmd-12-2285-2019, 2019.

Craig, J. R., Brown, G., Chlumsky, R., Jenkinson, R. W., Jost, G., Lee, K., Mai, J., Serrer, M., Sgro, N., Shafii, M., Snowdon, A. P., and Tolson, B. A.: Flexible watershed simulation with the Raven hydrological modelling framework, Environmental Modelling & Software, 129, 104 728, https://doi.org/10.1016/j.envsoft.2020.104728, 2020.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.

Feng, D., Liu, J., Lawson, K., and Shen, C.: Differentiable, learnable, regionalized process-based models with multiphysical outputs can approach state-of-the-art hydrologic prediction accuracy, Water Resources Research, 58, e2022WR032 404, https://doi.org/10.1029/2022WR032404, 2022.

Fenicia, F., Kavetski, D., and Savenije, H. H. G.: Elements of a flexible approach for conceptual hydrological modeling: 1. Motivation and theoretical development, Water Resources Research, 47, 2010WR010 174, https://doi.org/10.1029/2010WR010174, 2011.

Gupta, H. V., Kling, H., Yilmaz, K. K., and Martinez, G. F.: Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377, 80–91, https://doi.org/10.1016/j.jhydrol.2009.08.003, 2009.

Hansen, N. and Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies, Evolutionary Computation, 9, 159–195, https://doi.org/10.1162/106365601750190398, 2001.

Hargreaves, G. H.: Defining and Using Reference Evapotranspiration, Journal of Irrigation and Drainage Engineering, 120, 1132–1139, https://doi.org/10.1061/(ASCE)0733-9437(1994)120:6(1132), 1994.

Hascoet, T., Pellet, V., Oishi, S., and Miyoshi, T.: Differentiable river routing for end-to-end learning of hydrological processes, Journal of Geophysical Research: Machine Learning and Computation, 3, e2025JH000 760, https://doi.org/10.1029/2025JH000760, 2026.

Jing, X., Luo, J., Yang, M., Xiao, Y., and Yang, X.: dMoTpy v1.0.1: A Differentiable Library of 36 Conceptual Hydrological Models, Zenodo, https://doi.org/10.5281/zenodo.24110890, computer software, version 1.0.1, 2026.

Kavetski, D. and Fenicia, F.: Elements of a flexible approach for conceptual hydrological modeling: 2. Application and experimental insights, Water Resources Research, 47, 2011WR010 748, https://doi.org/10.1029/2011WR010748, 2011.

Kirchner, J. W.: Getting the right answers for the right reasons: Linking measurements, analyses, and models to advance the science of hydrology, Water Resources Research, 42, https://doi.org/10.1029/2005WR004362, 2006.

Knoben, W. J. M., Freer, J. E., Fowler, K. J. A., Peel, M. C., and Woods, R. A.: 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, 2463–2480, https://doi.org/10.5194/gmd-12-2463-2019, 2019.

Knoben, W. J. M., Freer, J. E., Peel, M. C., Fowler, K. J. A., and Woods, R. A.: A brief analysis of conceptual model structure uncertainty using 36 models and 559 catchments, Water Resources Research, 56, e2019WR025 975, https://doi.org/10.1029/2019WR025975, 2020.

Knoben, W. J. M., Raman, A., Gründemann, G. J., Kumar, M., Pietroniro, A., Shen, C., Song, Y., Thébault, C., van Werkhoven, K., Wood, A. W., and Clark, M. P.: Technical note: How many models do we need to simulate hydrologic processes across large geographical domains?, Hydrology and Earth System Sciences, 29, 2361–2375, https://doi.org/10.5194/hess-29-2361-2025, 2025.

Kratzert, F., Nearing, G., Addor, N., Erickson, T., Gauch, M., Gilon, O., Gudmundsson, L., Hassidim, A., Klotz, D., Nevo, S., Shalev, G., and Matias, Y.: Caravan - A global community dataset for large-sample hydrology, Scientific Data, 10, 61, https://doi.org/10.1038/s41597-023-01975-w, 2023.

Liang, X., Lettenmaier, D. P., Wood, E. F., and Burges, S. J.: A simple hydrologically based model of land surface water and energy fluxes for general circulation models, Journal of Geophysical Research: Atmospheres, 99, 14 415–14 428, https://doi.org/10.1029/94JD00483, 1994.

Lindström, G., Johansson, B., Persson, M., Gardelin, M., and Bergström, S.: Development and test of the distributed HBV-96 hydrological model, Journal of Hydrology, 201, 272–288, https://doi.org/10.1016/S0022-1694(97)00041-3, 1997.

Linke, S., Lehner, B., Ouellet Dallaire, C., Ariwi, J., Grill, G., Anand, M., Beames, P., Burchard-Levine, V., Maxwell, S., Moidu, H., Tan, F., and Thieme, M.: Global hydro-environmental sub-basin and river reach characteristics at high spatial resolution, Scientific Data, 6, 283, https://doi.org/10.1038/s41597-019-0300-6, 2019.

https://doi.org/10.5194/egusphere-2026-6870
Preprint. Discussion started: 20 November 2026
© Author(s) 2026. CC BY 4.0 License.



Loshchilov, I. and Hutter, F.: Decoupled Weight Decay Regularization, in: International Conference on Learning Representations, https://openreview.net/forum?id=Bkg6RiCqY7, 2019.

Newman, A. J., Clark, M. P., Sampson, K., Wood, A., Hay, L. E., Bock, A., Viger, R. J., Blodgett, D., Brekke, L., Arnold, J. R., Hopson, T., and Duan, Q.: Development of a large-sample watershed-scale hydrometeorological data set for the contiguous USA: data set characteristics and assessment of regional variability in hydrologic model performance, Hydrology and Earth System Sciences, 19, 209–223, https://doi.org/10.5194/hess-19-209-2015, 2015.

Ouyang, W., Ye, L., Chai, Y., Ma, H., Chu, J., Peng, Y., and Zhang, C.: A differentiable, physics-based hydrological model and its evaluation for data-limited basins, Journal of Hydrology, 649, 132 471, https://doi.org/10.1016/j.jhydrol.2024.132471, 2025.

Ouyang, W., Xu, S., Chai, Y., Zhuang, L., Liu, Z., Ye, L., Wu, X., Peng, Y., and Zhang, C.: A python framework for differentiable hydrological modeling and research workflow automation, Environmental Modelling & Software, 198, 106 895, https://doi.org/10.1016/j.envsoft.2026.106895, 2026.

Perrin, C., Michel, C., and Andréassian, V.: Improvement of a parsimonious model for streamflow simulation, Journal of Hydrology, 279, 275–289, https://doi.org/10.1016/S0022-1694(03)00225-7, 2003.

Samaniego, L., Kumar, R., and Attinger, S.: Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale, Water Resources Research, 46, 2008WR007 327, https://doi.org/10.1029/2008WR007327, 2010.

Shen, C., Appling, A. P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., Baity-Jesi, M., Fenicia, F., Kifer, D., Li, L., Liu, X., Ren, W., Zheng, Y., Harman, C. J., Clark, M., Farthing, M., Feng, D., Kumar, P., Aboelyazeed, D., Rahmani, F., Song, Y., Beck, H. E., Bindas, T., Dwivedi, D., Fang, K., Höge, M., Rackauckas, C., Mohanty, B., Roy, T., Xu, C., and Lawson, K.: Differentiable modelling to unify machine learning and physical models for geosciences, Nature Reviews Earth & Environment, 4, 552–567, https://doi.org/10.1038/s43017-023-00450-9, 2023.

Song, Y., Knoben, W. J. M., Clark, M. P., Feng, D., Lawson, K., Sawadekar, K., and Shen, C.: When ancient numerical demons meet physics-informed machine learning: adjoint-based gradients for implicit differentiable modeling, Hydrology and Earth System Sciences, 28, 3051–3077, https://doi.org/10.5194/hess-28-3051-2024, 2024.

Spieler, D. and Schütze, N.: Investigating the model hypothesis space: Benchmarking automatic model structure identification with a large model ensemble, Water Resources Research, 60, e2023WR036 199, https://doi.org/10.1029/2023WR036199, 2024.

Thébault, C., Knoben, W. J. M., Addor, N., Newman, A. J., Spieler, D., Vásquez, N. A., Song, Y., Gründemann, G. J., Carney, S., Kumar, M., van Werkhoven, K., Shen, C., Wood, A. W., and Clark, M. P.: Comparing multi-model mosaic and multi-model combination methods to simulate streamflow across the contiguous USA, Hydrology and Earth System Sciences, 30, 3945–3977, https://doi.org/10.5194/hess-30-3945-2026, 2026.

Trotter, L., Knoben, W. J. M., Fowler, K. J. A., Saft, M., and Peel, M. C.: Modular Assessment of Rainfall–Runoff Models Toolbox (MARRMoT) v2.1: an object-oriented implementation of 47 established hydrological models for improved speed and readability, Geoscientific Model Development, 15, 6359–6369, https://doi.org/10.5194/gmd-15-6359-2022, 2022.

Tsai, W.-P., Feng, D., Pan, M., Beck, H., Lawson, K., Yang, Y., Liu, J., and Shen, C.: From calibration to parameter learning: Harnessing the scaling effects of big data in geoscientific modeling, Nature Communications, 12, 5988, https://doi.org/10.1038/s41467-021-26107-z, 2021.