Technical corrections after the report of 29-04-2020

- We clarified that actual evaporation is required by the concept in Section 2.1.
- We rewritten the first paragraph of Section 3.2 to avoid confusion.
- The dashed green line is indeed not visible due to overlay, we have added this to the caption.

Point-by-point response to reviews

Below, we will give a point-by-point response the reviews, with the comments from the reviews in black, and our reply in blue.

Response to AR1

Dear authors, thank you for submitting your paper. In the following, you will find my comments, suggestions, and opinion of your contribution.

Summary

The paper "A distributed simple dynamical systems approach (dS2 v1.0) for computationally efficient hydrological modelling" is about a distributed hydrological modelling based on the simple dynamical system approach, SDS (Kirchner, 2009). The model is grid-based (cell of 1km2) and enables rainfall-runoff simulations for mesoscale basins (up to 2000 km2) at high temporal and spatial resolution with a new 3-parameter discharge sensitivity function. The model is first developed and assessed for synthetic experiment and then applied to the alpine Thur catchment (1700 km2). Authors used Monte-Carlo methodology to generate model parameter sets constrained within the synthetic parameter ranges. The model is complemented with snow and routing modules to take into account the lag-time.

General comments

The manuscript is clearly written but has in its present form some major defaults. The authors put forward the novelty of the proposed approach. This is not right since the approach is not unique in its formulation. What is proposed is a simple dynamical system approach by Kirchner applied in its distributed form. Such model, however already exists (please check Adamovic et al. 2016, Journal of Hydrology). The latter was not cited in this article. Adamovic et al. (2016) implemented the SDS approach on Ardeche catchment (around 2400 km2) and developed the model called SIMPLEFLOOD that is computationally fast and coupled with a kinematic wave flow propagation module.

We understand the concern from AR1 about the fact that we did not include a reference to SIMPLEFLOOD. In this new version of the manuscript, we have added a paragraph on SIMPLEFLOOD in the introduction. We have added reasoning on why we believe that dS2 is an important addition to the existing suite of hydrological models is still a valuable addition as a conceptual model (distributed using a grid, focus on computational efficiency, difference in numerical implementation, inclusion of additional processes such as snow for application to colder climates and/or more mountainous areas, and open source code).

Specific comments

Here are, however, my short comments that I think can contribute to its better content once reformulating and updating the article.

Page 2. Line: 25 The authors mention continental-scale forcing datasets in relation with merged radar data, interpolated station data, and atmospheric reanalysis. It would be valuable for readers if they know in more details about actual spatial but also temporal scales of provided references.

After the references, we have added a spatial and temporal scales on which these datasets are available.

Page 2. Line: 35 The phrase "An efficient distributed conceptual model to tackle these kinds of issues is currently lacking." is not correct. Adamovic et al. (JoH, 2016) already developed this approach at an hourly time step and applied it to the Ardeche catchment.

See our reply above under "General comments".

Page 3. Line: 25 Similar to the previous comment. The approach was already applied to the mesoscale catchment of 2400 km2 (Adamovic et al., 2016).

See our reply above under "General comments".

Page 6. Line: 15 The introduction of a third parameter that avoids having a curvature that diverges computations is interesting even though it has its limitations since it seems that three parameters depend on each other.

It does indeed add some complexity to the two already correlated parameters α and β. As with any fitting of any curve, the fitting parameters will not be completely independent. This is not a specific disadvantage of the curve chosen. However, given that most previous studies showed a sensitivity function which could not be captured with a simple power–law, we believe that this is one of the most elegant solutions to cover known catchment behaviour.

Page 8. Flow routing Authors introduced simplified routing module that is not explicit but rather based on a simple temporal delay to the outlet. More sophisticated flow routing module is described in Adamovic et al. (2016) which takes into account the feedback effects.

See our reply above under "General comments". We present a conceptual model. Similar to the assumption of a single subsurface storage, we make a simplifying assumption that flow speed is constant across the stream network. We are fully aware of the fact that this is a crude assumption, but it is one that works well in terms of both model performance and computational efficiency. This concept is not new (width function, see Kirkby (1976)), and has been used in other studies as well (e.g. Franchini and O'Connell, 1996).

Page 12. Closure of the water balance In the model, the water balance seems closed only in conditions when AET=PET (humid conditions). If the model is to be applied on pan-European scales with different conditions, the question about the AET estimation should be addressed.

AR1 is correct that dS2 is currently not able to reduce evaporation during e.g. dry periods. This is why we rely on the user to provide accurate actual evaporation input data, as is stated in the Discussion. These datasets are now increasingly becoming available at temporal and spatial resolutions that dS2 runs at, so we think dS2 should be seen as a platform that can directly utilize this kind of data for runoff simulations. This approach (relying on input of AET rather than PET) is indeed fundamentally different from current modelling approaches.

Page 13. Parameter sensitivity It would be useful for a reader to know what the realistic parameter ranges are and how did you choose them since you applied the method on the synthetic catchment in this section.

The ranges presented in this figure show realistic parameter ranges, based on comparisons between the shape of the resulting discharge sensitivity function with previous studies using the simple dynamical systems approach. This information was indeed lacking, but is now added to the manuscript in the section about parameter sensitivity.

Page 16. Example application Line: 23. Did you also run the MC analysis with the fewer or higher number of iterations? Why did you choose 25000? Many authors used fewer iterations for higher-order parametric models (e.g. Tekleab et al. 2011).

We selected this number of runs to ensure good coverage of the parameter space. Initially the number was based on the study by Melsen et al. (2016a). As VIC is a higher-order parametric model, this number of runs should be sufficient for a model like dS2. We do not see a problem when using too many runs for a MC analysis, problems could indeed arise when one uses a too small sample size.

Page 17. Line: 7 "Parameter set that performer best for the event". Is it the one that showed the highest KGE? If it is the case, what was the value?

This was indeed not correct in the previous version of the manuscript. We have updated the figure to ensure the correct runs are shown. The runs selected for the event are the runs with the best KGE performance over the total period showed in the other panels, where also the KGE values of these runs are shown.

Page 17. Line: 9 ". . .the timing of the peak is well simulated in all three basins". It seems that for the Andelfingen catchment (Fig. 10e), there is a peak delay of around 6 h between the observation and simulation for this specific event. I would suggest reformulating this sentence and discussing what could be a reason for such peak delay since it is the largest explored catchment. Could it be that the simplified routing module shows the lack of performance in this case? Testing it in more catchments could give a more robust answer.

The text and the figures were not updated correctly in the previous version of the manuscript. In this new version of the manuscript, we have updated both the figure and text to better represent the output of the model. With the new run, the discrepancy in timing of the peak between observation and simulation is reduced. Since this basin contains many sub-basins,

we decided to keep this case study to only a single basin. We are currently applying the model to different basins to investigate the performance.

Technical corrections

Figure 1. Panels in capitals A, B, C instead of small letters. This will improve the text visibility.

To comply with the standards of GMD, we have decided to keep the panels in lower case letters.

Figure 10 a. I would suggest colouring the catchments for better visibility.

We have highlighted the shape of the three catchments shown in this figure.

Figure 10 b, c and d. I would suggest plotting these figures on a log-scale for better visibility and assessing the influence of evapotranspiration.

We tried plotting these panels on a log-scale, but this gave a distorted look on the model output, as much less emphasis is placed on the peaks. Since the aim of this figure is to visualize the model output, we decided to keep this figure with a normal axis.

Figure 10.e I would suggest not using the light colours for observations since it is not well readable (i.e. Mogelsberg catchment). Asterix or other dash types could be more appropriate in this case.

We have updated the figure and changed the linestyle of the observations to improve visibility.

Response to AR2

Note: some of our answers are largely copied from our reply to AR2 in the discussion.

General comments

Summary The manuscript A distributed simple dynamical systems approach (dS2 v1.0) for computationally efficient hydrological modelling by Buitink and colleagues introduces a distributed hydrological conceptual model oriented at the simple dynamical systems approach by Kirchner (2009). The model has been designed to simulate rainfall–runoff dynamics with high spatial and temporal resolution at small (100 km2) up to mesoscale (103 km2) catchments. The model should be straightforward to apply, computationally efficient, and the associated Python code is openly available and easy to understand and modify. In their manuscript the authors present a sensitivity analysis of model parameters and apply the model in the alpine Thur catchment. Most parts of the paper are well structured and good to understand. However, I see some serious problems with the paper (and the model) that should be resolved before final publication. Even more than for my concerns regarding originality and efficiency this holds true for the applicability of the model, as I will explain in the following paragraphs.

Originality As even the authors admit, many different conceptual hydrological models already exist. Therefore the question arises if we really need yet another model. My personal answer would be: maybe, if we can learn something from it. However, dS2 seems to me rather like a typical black box model and I don't see a way to learn anything about processes or the catchment. Even though the authors see their model as a valuable tool for educational purposes (page 20, line 17 ff.), other conceptual models based on multiple storages (or buckets) are, in my opinion, much more suitable for education. For instance, storages can be related to natural phenomena (e.g. surface, quick subsurface, and baseflow) and students can learn something about processes and parameter sensitivities (for an illustration, see e.g. Fenicia et al., 2016). As far as I can tell, dS2 serves solely as a very simple rainfall–runoff transformation tool and is therefore nothing new. I wonder how we can learn something about processes in a catchment, as the authors claim. To me this is not evident from the example application.

Whether or not the hydrological community needs more models has been a long debated topic. We should stress that we would not have developed the models without the belief that it fills an important niche. There are a few important things that make this model stand out from the already existing conceptual models, and we believe that we can learn from the model.

First of all, the underlying concept of the dS2 model, the simple dynamical systems (SDS) approach, is based directly on discharge observations (Kirchner, 2009), as is stated in the introduction. The SDS approach is popular and sparked many scientific studies/discussions: the original paper is cited over 300 times according to Web of Science. The dS2 model allows us to apply a proven concept in a way that it supports larger catchments with more spatial variability. In contrast to this, the many bucket based conceptual models are based on our conceptual understanding of the hydrological system rather than observations. Therefore, dS2 relies on a different philosophy than the vast majority of conceptual models. Additionally, the

SDS approach calculates changes in storage only expressed in terms of discharge rather than absolute storage values. As this is different from the large majority of conceptual models, dS2 can give new insights into the hydrological response.

Furthermore, as this is a computationally efficient hydrological model (though we do not state that this is the most efficient hydrological model, see our reply below in the "Efficiency" section), it allows for relatively cheap sensitivity uncertainty and sensitivity studies. This is also something the majority of existing conceptual models cannot achieve.

Another point made by AR2 is that dS2 is solely a rainfall-runoff transformation tool. Simply stating, this is correct, but the same point can be made for every model — when discharge is the main variable of interest. Furthermore, the term rainfall–runoff transformation tool can be perceived as something 'black box', while there is definitely a physical hydrological understanding underlying the concept of this model. This ties directly to another point made by AR2: that this model will not help understanding processes within a catchment. This is correct, as these processes are all indirectly captured by the sensitivity function. However, the model can definitely help with understanding how the discharge of a catchment will respond under different scenarios, and how this relates to storage in a catchment. It can also help to investigate whether runoff dynamics in a particular catchment are consistent with those from a simple dynamical system. We have made some changes in the introduction to better state the aim of this model.

In order to more explicitly state the niche of this model, we have also slightly altered the title of the manuscript to: A distributed simple dynamical systems approach (dS2 v1.0) for computationally efficient hydrological modelling at high spatio-temporal resolution.

Efficiency The authors stress at many occasions the computational efficiency of their model. On the other hand, computational efficiency has been sacrificed in favour of code readability by implementing it in Python, a widely used scripting language that is known for its well structured syntax in contrast to much more efficient compiled languages such as C++ or Fortran. Therefore, in my view the argument of computational efficiency, which is even advertised in the title of the manuscript, does not hold. A good compromise would have been to outsource the most expensive parts of code to a compiled language and use Python as an interface, as has been done for other models (e.g. TOPMODEL, for which an R interface exists, while the core model is written in Fortran, see Metcalfe, Beven and Freer, 2015)

AR2 correctly stated that we sacrificed some computational efficiency to ensure code readability. Maybe the word "efficiency" in the title suggests that we tried to build the most computationally efficient conceptual hydrological model, but this was not our main goal. If this were the case, it would indeed make a lot more sense to use compiled languages such as C++ or Fortran. Of course, we do wanted to utilize the characteristics of the SDS approach in order to have a model that is computationally efficient. Within the flexible and widely-used Python programming language, we tried to write the code as efficient as possible. Therefore, the model heavily depends on the Numpy library, which utilizes C libraries for its calculations. This library allows for example the vectorization of functions, something that is not supported by the default Python functions.

Furthermore, as Python is continuously getting more and more popular, it allows other users to understand, improve and/or extend the model, or change part of the code depending on the research question at hand. We see models not as static entities, but rather as flexible environments in which ALL elements (not just parametrizations but also the numerical "core") can easily be adapted. A widely known language such as Python is much better tailored to this task . This could indeed also be done with using Python as an interface, but given the previous arguments we have chosen Python as the preferred programming language. We have added this reasoning to Section 3.1 to better explain our reasoning behind choosing Python as the programming language.

Applicability I acknowledge that the code of the model is indeed well readable, even for persons with little Python experience (including me). I only wonder how I should apply the model. The authors present an example application in their manuscript but no example data to test the model. What is even more, the input and output file structures are not well explained, neither in their submitted manuscript nor in the assets or the github repository. In that way I don't know how to initialise and apply the model. This is probably the most severe flaw of the presented work. In my opinion it is also violating the core principles of GMD's model code and data policy.

We have created a brief model manual explaining the required model input and data formats. This manual includes pieces of code to show how one can setup, run, and analyse output of dS2. The manual can be found in the GitHub repository of the dS2 model (https://github.com/JoostBuitink/dS2/tree/master/example_application) and will extended and be kept up to date with possible future changes to the model.

Sensitivity analysis I see a problem with the Sobol' method for variance-based sensitivity analysis that the authors applied. This method may not be applied in case parameters are correlated (which is the case, as is stated in the text). I will provide some more reasoning and suggestions in the specific comments.

We are aware that the Sobol' method works optimally in the case where parameter are not correlated. This is not exactly the case in our model, yet we still decide to use this method since it is the most widely used method to investigate the parameter sensitivity. Furthermore, the aim of this sensitivity analysis is to give a first impression of the influence of the parameters on performance metrics. To give an idea of the effect of parameter interaction, we also show the "total effect", but will focus on the "main effect" for this study.

Specific comments

In the following I will provide some specific comments. Quotations from the paper appear as emphasised text. Specific passages in the text are referenced as pmLn, meaning page m, line n.

- "There is a growing need for easy-to-apply models that can utilize the potential of spatially distributed input data" (p2L26–27). Many (most?) models already utilize spatially distributed information. I also think the demand for easy-to-apply models always has been present (whereas the judgement of what is easy-to-apply is rather subjective).

  We have added a spatial and temporal resolution to this paragraph to clearly state that we aim at data at both high spatial and temporal resolutions. With the addition of the manual, the model should be easy–to–apply.

- "water is most often transferred to the outlet as a post-processing function. This is, however, not necessarily the most computationally efficient way to deal with spatially distributed data" (p2L29–30). And what is the most (or at least more) efficient approach? Isn't the dS2 model essentially doing the same or what exactly is the difference?

  The sentence was referring to the first part of the sentence before that. We have rephrased this part to clearly state what we mean with the "most computationally efficient way."

- "Many aspects of distributed modelling [...] require a high number of runs which further increases the computational demand. An efficient distributed conceptual model to tackle these kind of issues is currently lacking." (p2L34–35). With computational infrastructure nowadays (HPCs, cloud computing) it is already possible to conduct hundred thousands or even millions of iterations of conceptual models within acceptable time frames, depending on resolution and spatio-temporal domain (see examples in Beven and Binley, 2014). Why then do we need more computationally efficient conceptual models? To conduct billions of runs in the same time? I doubt that this would substantially increase the value of uncertainty analysis and parameter estimation.

  This is correct, yet a model like dS2 allows users to do these computationally demanding on their own machines, rather than using HPCs (and the costs related to that). We have rephrased this part to clarify our point.

- "Low-level languages generally perform faster calculations, but this comes at the price of user-friendliness and ease-of-use." (p9L3–4). I don't understand this relationship. Doesn't user-friendliness and ease-of-use depend on the interface between model and user, i.e. the structure of input and output files, how to run the model etc.? You can also program complicated models with high-level languages. And what about users, who prefer GUIs? For them a purely command line-based model will never be considered user-friendly. Moreover, user-friendly models, in my opinion, always contain a good explanation of file structures and a simple test case, which is unfortunately not the case for dS2.

  We have rewritten this part to clearly argue why we believe that Python is a solid programming language for this model. Together with the new model manual, it is easier for users to apply this model in their own research areas.

- "Simulating Europe for three months at hourly time steps and at a resolution of 5x5 km2" (p9L18). Does it make sense to do that with dS2? Otherwise I don't understand why you choose such an unrealistic comparison.

  We added this value as an reference, as this is likely to speak more to the reader than just the number of pixels. We have slightly rephrased the sentence to make this clear.

– Section 3.2 (Adaptive time stepping): I think this sub-section can be hard to follow for anyone who is not familiar with numerical integration. E.g. what is the Runge-Kutta scheme, what are fourth and fifth order estimations? Maybe you can add a paragraph with a (very) short introduction to numerical integration, how it basically works, and why it is needed. It might also be worthwhile to stress the relevance in hydrological modelling as the issue is often neglected (most hydrological models just use explicit Euler with operator splitting). Some interesting papers about the topic: Clark and Kavetski (2010), Kavetski and Clark (2010), Kavetski and Clark (2011) and Schoups et al. (2010).

  We have added a brief introduction on numerical integration in Section 3.2, and how our approach compares to common approaches in hydrological modelling.

– Comment on numerical integration: As the dS2 is so extremely computationally efficient I wonder why you don't try the much more accurate implicit solvers. Of course they will increase computation time, but on the other hand will deliver much more accurate results (and potentially more reliable parameter and uncertainty estimations, see e.g. Kavetski and Clark, 2011). Maybe the GNU Scientific Library (https://www.gnu.org/software/gsl/) is worth a try (never tried by myself, but the website says the interface was designed to be simple to link into very high-level languages, such as GNU Guile or Python).

  This could be a possibility, yet we believe that after extensive (stress) testing our custom numerical solver is able to accurately and efficiently solve the differential equation. One could opt for more complex and therefore more computationally demanding solvers, but question arises whether this is worth the additional computational demand.

– "[...] base the volume estimation on the mean discharge of the resulting shorter steps" (p13L2). Is that indicated by Qinternal in Fig. 7? Please clarify that in the figure, as Qinternal itself is not explicitly defined.

  We have clarified the meaning of Qinternal, both in the text and in the caption.

– "the current version of the model only outputs discharge at the end of the time step" (p13L8–9). But internally Qinternal is used, i.e. $\int_{t-1}^{t} Q_{internal}$ to calculate St?

  This is not correct, and we have clarified this in the text in Section 3.3.

– Fig. 8: In the text you write response of the model to each parameter, but the model contains more than the five parameters shown in Fig. 8?! Please be more explicit about what you mean.

  We have explicitly stated which parameters we are investigating, as the majority of parameters mostly influence input/output operations and numerical stability.

– Section 4 (Parameter sensitivity, Sobol' sensitivity analysis): The Sobol' method requires that parameters are independent but in relation to Fig. 8 it is mentioned that some parameters are correlated. This will distort the results of the sensitivity analysis. Or produce strange results as can be seen in Fig. 8, namely that in some cases the total effect is smaller than the main effect (or does the size of the bars reflect main + total effect? Please clarify). It seems also strange that for KGE – β the total effect is always zero (if my interpretation of the bars is right). In any case, a possible workaround for correlated parameters is presented by Kucherenko, Tarantola and Annoni (2012) (not really an ad-hoc implementation). However, to see if the effort is really necessary, please first check the correlations among parameters (e.g. via covariance matrix). You might also consider a different method for sensitivity analysis, which is not affected by correlations (see review papers for guidelines, e.g. Pianosi et al., 2016).

  As mentioned above, this analysis is performed to give an impression of the parameter sensitivity, hence why we also focus on the main effect, and not on the total effect. We have also added the limitations of this method with respect to the correlated parameters of dS2.

– "These graphs also show that there are some parameter correlations influencing the results" (p16L1). What exactly do you mean? The discrepancy between main and total effects I mentioned earlier?

  We did indeed mean this discrepancy, and we have clarified this in the text.

– "We see that, although the magnitude of the peak is not fully captured in the Rietholzbach and Andelfingen basins, the timing of the peak is well simulated in all three basins." (p17L8–9). But for Andelfingen the simulated peak occurs several hours before the measured peak, even with the routing module (the difference between routing and no routing module seems to be less than the difference between observations and routing module). As I see, this specific issue has also been addressed by the other reviewer

We have updated both the text and the figure as they were old (incorrect) version. This new figure and explanation in the text fixes and addresses the issues raised.

Technical corrections

We have implemented all technical corrections made by AR2.

List of relevant changes

- The introduction is slightly altered so the goal of dS2 is more clearly stated. The introduction also highlights the SIMPLEFLOOD model to and states how dS2 is different from SIMPLEFLOOD.
- The reasoning and implications of the chosen programming language is now more clearly formulated.
- We have added a paragraph which shortly explains the importance of numerical solvers, including a brief description of higher order numerical solvers.
- A model manual was previously missing: we have created a brief model guide to explain what input data is required to setup the model, and how to run the model and analyse the output. This manual will be kept up-to-date on the GitHub page of the model (https://github.com/JoostBuitink/dS2/tree/master/example_application).
- We have slightly altered the title to better state the gap dS2 tries to fill.
- We have improved Figure 9, to better depict the difference between the main and total effects.
- We have improved Figure 10 by: highlighting the basins of interest, correcting the model runs shown in panel (e), adjusting the linestyle in panel (e) to make the observations better visible, and improved the caption of the figure.

A distributed simple dynamical systems approach (dS2 v1.0) for computationally efficient hydrological modelling at high spatio-temporal resolution

Joost Buitink[1], Lieke A. Melsen[1], James W. Kirchner[2, 3, 4], and Adriaan J. Teuling[1]

[1]Hydrology and Quantitative Water Management Group, Wageningen University, Wageningen, The Netherlands
[2]Department of Environmental Systems Science, ETH Zurich, Zurich, 8092, Switzerland
[3]Swiss Federal Research Institute WSL, Birmensdorf, 8903, Switzerland
[4]Department of Earth and Planetary Science, University of California, Berkeley, California, 94720, USA

Correspondence: Joost Buitink (joost.buitink@wur.nl)

Abstract. In this paper, we introduce a new numerically robust distributed rainfall runoff model for computationally ~~efficiency~~ efficient simulation at high ~~(hourly) temporal~~ spatio-temporal resolution: the distributed simple dynamical systems (dS2) model. The model is based on the simple dynamical systems approach as proposed by Kirchner (2009), and the distributed implementation allows for spatial heterogeneity in the parameters and/or model forcing fields at high spatio-temporal resolution (for instance as derived from precipitation radar data). The concept is extended with snow and routing modules, where the latter transports water from each pixel to the catchment outlet. The sensitivity function, which links changes in storage to changes in discharge, is implemented by a new 3-parameter equation that is able to represent the widely observed downward curvature in ~~log-log~~ log–log space. The simplicity of the underlying concept allows the model to calculate discharge in a computationally efficient manner, even at high temporal and spatial resolution, while maintaining proven model performance~~at high temporal and spatial resolution~~. The model code is written in Python in order to be easily readable and adjustable while maintaining computational efficiency. Since this model has short run times, it allows for extended sensitivity and uncertainty studies with relatively low computational costs. A test application shows a good and constant model performance across scales ranging from 3 to over 1700 km^2.

1 Introduction

Hydrological models are essential tools for applications ranging from sensitivity analysis to impact assessment and forecasting. Generally, the aim of ~~rainfall-runoff~~ rainfall–runoff models is to simulate streamflow given ~~a precipitation time series~~precipitation input. Depending on factors such as the research aim and the climatological/geological setting, different model structures or process ~~representation~~ representations might be preferred. This, in combination with the inherent complexity and heterogeneity of (sub)surface hydrological processes, has ~~lead~~ led to the development of numerous different hydrological models over the past decades, each with ~~their~~ its own focus. Examples of such ~~rainfall-runoff~~ rainfall–runoff models and modelling tools include, amongst many others: SWAT (Arnold et al., 1998), HBV (Lindström et al., 1997), TOPMODEL (Beven and Kirkby, 1979), VIC (Liang et al., 1994, 1996), SPHY (Terink et al., 2015), FUSE (Clark et al., 2008), SUMMA

(Clark et al., 2015), PCR-GLOBWB (Sutanudjaja et al., 2018) and WALRUS (Brauer et al., 2014). Although there is an ongoing debate about whether the hydrological modelling community should move towards a community model (Weiler and Beven, 2015), different models representing a wide range of complexity and different process representations might be necessary to adequately characterise uncertainty.

Hydrological models are often classified from more conceptual to more process-based models. Conceptual models have fewer processes explicitly ~~parameterized~~parametrized, and as a result their limited number of parameters makes them easier to calibrate. In conceptual models, catchments are often represented by a series of buckets or storages, which mimic processes with different response times. Their typical scale of application is that of small to medium (mesoscale) catchments, generally in a lumped fashion. Process-based models, on the other hand, contain ~~a much higher number of~~ many more parameters. These models are often applied in a distributed fashion, where many of the parameter values are based on maps of vegetation and soil properties. However, often conceptual parameters remain that require calibration or tweaking. Even though process-based models should give a better representation of the physical reality, many models can easily be beaten in performance by a simple neural network (Abramowitz, 2005), or model results can be reproduced without much loss of accuracy by models with a much ~~lower~~ smaller number of parameters (Koster and Suarez, 2001; Best et al., 2015; Liu et al., 2018). So if the research aim does not require the use of complex or specific process representation, simple models will likely outperform more complex models in many applications.

Whereas conceptual models often perform ~~satisfactory~~ satisfactorily at the daily resolution at the lumped basin scale, they lack the ability to explicitly simulate spatially distributed processes that might be needed to accurately predict streamflow response at larger scales. In a study over a large number of basins in France, Lobligeois et al. (2014) found that (hourly) model performance markedly increased from the (lumped) basin-scale to a resolution of around 10 km^2 when using aggregated radar precipitation as model input. Other studies (e.g. Ruiz-Villanueva et al., 2012) have also highlighted the importance of spatial variability of rainfall, in particular the movement of storms with respect to the channel network, for flash flood simulation. Another example of a spatially distributed process that affects streamflow is the melting of snow that can depend both on elevation (via temperature) and aspect (via radiation). Comola et al. (2015), for example, showed that aspect needs to be considered for accurate simulation of snowmelt and runoff dynamics in mesoscale catchments. The spatial organization of the stream network within a basin also affects the response to rainfall, as has been shown by studies based on the catchment width function or the Geomorphological Instantaneous Unit Hydrograph (Kirkby, 1976; Rodríguez-Iturbe and Valdes, 1979). Thus, a spatial resolution in the order of 1–5 km to capture effects of rainfall variability and stream network organization, combined with a temporal resolution of 1 h or less to capture individual (convective) rain storms, is necessary for realistic rainfall-runoff modelling at larger scales.

The need to improve spatially explicit information in hydrological models aligns with the increasing availability of high-resolution continental-scale forcing datasets. These include for instance merged radar data ~~(Huuskonen et al., 2013)~~(e.g Huuskonen et al., 2 , interpolated station data ~~(Cornes et al., 2018)~~(e.g. van Osnabrugge et al., 2017; Cornes et al., 2018; van Osnabrugge et al., 2019) , or atmospheric reanalysis ~~(Albergel et al., 2018).~~ products (e.g Albergel et al., 2018). These datasets have a spatial resolution in the order of one kilometre and a temporal resolution in the order of one hour. There is a growing need for ~~easy-to-apply~~

easy–to–apply and easy–to–adjust models that can ~~utilize~~ exploit the potential of spatially distributed input data at high spatial and temporal resolution.

Conceptual models have been applied in a (semi-)distributed manner to account for spatially distributed input data: a lumped model is applied ~~for~~ at each individual grid cell, and water is most often transferred to the outlet ~~as~~ using a post-processing function ~~. This~~ that accounts for catchment routing. This method of running the model subsequently for each individual grid cell is, however, not necessarily the most computationally efficient way to deal with spatially distributed data, but ~~rather~~ often the result of historical developments. Whereas increased computational power has driven the application of distributed models at increasingly fine (spatial) resolution (Melsen et al., 2016b), it ~~also lead~~ has also led to new challenges: ~~Many~~ many aspects of distributed modelling, such as uncertainty estimation and (spatial) parameter estimation, require a ~~high~~ large number of runs which further increases the computational demand. ~~An efficient distributed conceptual model~~ This demand is in the vast majority of cases too high to be tackled by individual computers, and thus computational infrastructure such as high performance clusters are used. However, there are costs related to this procedure, while the "free" computational power in the individual computer could be used more optimally. A conceptual model that can deal with high resolution gridded data to tackle these kind of issues in an efficient manner is currently lacking.

An example of a conceptual model that, in spite of its extreme simplicity, has shown ~~a~~ good performance for discharge simulation at the scale of smaller catchments and at fine temporal (hourly) resolution is the simple dynamical systems (SDS) approach introduced by Kirchner (2009). This concept is based on the assumption that discharge is solely dependent on the total amount of stored water in a catchment. It translates changes in storage to changes in discharge using a discharge sensitivity function, without describing internal catchment processes. This sensitivity function is typically ~~parameterized~~ parametrized by a 2-parameter ~~power-law~~powerlaw, however several studies have suggested a more complex downward-curving rather than linear behaviour in double-logarithmic space ~~(Kirchner, 2009; Teuling et al., 2010; Adamovic et al., 2015)~~ (Kirchner, 2009; Teuling et al., 2010; Adamovic et al., 2015). Moreover, the system parameters can be inferred from streamflow recession analysis and potentially from hillslope characteristics (Troch et al., 2003), potentially removing the need for model calibration (Melsen et al., 2014). The model's simplicity has the important advantages that discharge can be simulated based on a single equation — which can easily be vectorized for distributed implementation ~~, and~~ — and that the model only needs to store a limited number of variables, reducing the model output fields — since storage and discharge are directly linked via the discharge sensitivity function~~they don't need to be stored separately thus reducing the number of model output fields~~. Especially this latter part is fundamentally different from the many "bucket-based" conceptual models, which use storage in conceptual reservoirs to determine the outflow. In the original test in the humid Plynlimon catchments (area 8.70 and 10.55 km^2), Kirchner (2009) found ~~Nash-Sutcliffe~~ Nash–Sutcliffe Efficiencies (NSE's) exceeding 0.95 during the model validation when calibrating model parameters, and efficiencies exceeding 0.90 when parameters were obtained from recession analysis. Others have also found the concept to work well in less humid catchments. Teuling et al. (2010) found the method to generally work well in the small (3.3 km^2) Swiss Rietholzbach catchment. Although the method can be expected to work best in hilly catchments, Brauer et al. (2013) found the model to produce reasonable efficiencies after calibration in the small (6.5 km^2) Dutch Hupsel Brook catchment. Adamovic et al. (2015) reported NSE values exceeding 0.6 for most

years in several Ardechian catchments in the order of 10–100 km^2. Given the simplicity and good performance of the simple dynamical systems approach at the spatial (order 10 km^2) and temporal (1 h) resolution required for optimal simulation of ~~rainfall-runoff processes~~ runoff in larger (mesoscale) catchments, combining it with simple representations of routing and snowmelt should result in a model that satisfies the required criteria outlined in the previous paragraphs. In a first approach to apply the simple dynamical systems approach spatially, Adamovic et al. (2016) developed a semi-distributed implementation of the SDS approach (SIMPLEFLOOD), only using sub-basins to distribute the catchment and not a grid. This model, however, was not optimized for computational efficiency and/or numerical stability over a wide range of model parameters required for optimization studies, and the code was not made available as open source.

Here, we present a flexible and computationally efficient distributed implementation and extension of the simple dynamical systems approach that can be used to investigate the spatially distributed hydrological ~~processes in a flexible and efficient manner~~response using data at high spatial and temporal resolution: the distributed simple dynamical systems (dS2) model. Our aim was to create a model that is ~~able to computationally efficient simulate~~ capable of computationally efficient simulations of discharge in mesoscale basins at high ~~temporal and spatial~~ spatio–temporal resolutions, but also to develop a model code that is easy to use, read and modify. Therefore, the model is written in Python, ~~sacrificing some calculation efficiency.~~ although there are more computationally efficient languages available. We believe, however, that this is the optimum between model speed and code adjustability required in most hydrological model studies. In this model, the catchment is divided into smaller sections using a regular grid, and discharge is simulated according to the SDS approach for each pixel. This distributed implementation allows the concept to be applied to bigger catchments, and also allows for spatial heterogeneity, both in the forcing and in the parameters. Since the original concept consists essentially of only one differential equation, it can be applied in a computationally efficient fashion, vectorizing all cells in the catchment. Snow and routing modules are added to the model to allow for application in snow-dominated regions, and to transport the water from each pixel to the catchment outlet via the drainage network. This efficient distributed implementation lowers the computational burden for high spatial and/or temporal resolution studies, and opens doors for extensive uncertainty studies. We will first introduce the model concept and describe the technical application. ~~Then~~ After that, we discuss the parameter sensitivity and finally, we show an application of the model.

2 Model concept

2.1 Simple dynamical systems approach

The simple dynamical systems approach proposed by Kirchner (2009) combines the ~~conservation-of-mass equation~~ conservation–of–mass equation (assuming a constant density) with the assumption that discharge is solely dependent on the total storage (excluding snow and ice) in the area of interest:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = P - E - Q, \tag{1}$$

$$Q = f(S), \tag{2}$$

where S ~~represent~~ represents the total storage, P the precipitation, E the ~~(actual)~~ actual evaporation and Q the discharge. Differentiating Eq. (2) and combining with Eq. (1) results in the following equation:

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = \frac{\mathrm{d}Q}{\mathrm{d}S}\frac{\mathrm{d}S}{\mathrm{d}t} = \frac{\mathrm{d}Q}{\mathrm{d}S}(P - E - Q) = g(Q)(P - E - Q), \tag{3}$$

where we define $\frac{\mathrm{d}Q}{\mathrm{d}S}$ as the sensitivity function $g(Q)$, describing the sensitivity of discharge to changes in storage. ~~We introduce an evaporation reduction parameter ϵ, which acts as a simple translation from potential to actual evaporation ($E_{\mathrm{act}} = \epsilon \cdot E_{\mathrm{pot}}$).~~ Note that the evaporation ~~here represents the actual evaporation~~represents actual evaporation, as the concept does not ~~directly allow for~~ simulate evaporation reduction as result of e.g. soil moisture stress ~~. A simple evaporation~~ during dry periods. Ideally, a soil moisture model would provide the correct actual evaporation values, or when applying the concept to humid areas, the actual evaporation is equal to a factor times the potential evaporation. To support the latter, we introduce the evaporation correction parameter ϵ, which can correct the provided (potential) evaporation input data ($E = \epsilon \cdot E_{\mathrm{input}}$). Additionally, to prevent numerical issues, a simple evaporation reduction switch is added, which is described in Section 3.2.

Originally, the SDS approach was introduced as a lumped approach to simulate discharge in small catchments of approximately 10 km^2 (Kirchner, 2009). Several studies have subsequently applied this concept to other catchments in Europe (Teuling et al., 2010; Krier et al., 2012; Brauer et al., 2013; Melsen et al., 2014; Adamovic et al., 2015). Some of the catchments in these studies had a size similar to the original scale from Kirchner (2009), yet it was also applied to catchments up to 1000 km^2 in size. One could argue whether the concept is still valid at a scale so different from the scale for which it was initially developed (Beven, 1989, 2001; Sivapalan, 2006; McDonnell et al., 2007), and whether a single sensitivity function is sufficient to capture the spatial complexity of substantially larger basins. In regions with high spatial heterogeneity, grid-based models are likely to yield more realistic results than lumped models (Lobligeois et al., 2014).

To respect the original scale of development and to capture spatial variability, we have developed a distributed implementation of the simple dynamical systems approach. Our distributed implementation builds on the original concept as proposed by Kirchner (2009), and extends this concept with simple snow and routing modules. For the distributed implementation, we assume that the SDS approach is valid for each pixel of a rectangular grid. By defining pixels with a size corresponding to the original scale (in the order of 1 km^2), the scale of application remains similar to the original scale, and both the forcing and the model parameters can be defined for each individual pixel (see Figure 1). In this distributed implementation, we allow precipitation to fall as snow~~,~~; see Section 2.3. We added a routing module to transport water from each pixel to the river outlet, by adding a time delay to each pixel based on the distance to the outlet and a travel speed parameter~~,~~; see Section 2.4. The model can be run with different choices for Δt, yet in order to respect the ~~spatio-temporal~~ spatio–temporal resolution the default time step is one hour. This model has been built with a focus on computational efficiency, meaning that all grid cells in the catchment are stored in a single vector, allowing for vectorized computations (see Fig. 1b). This results in a matrix with the rows and columns indicating time and space, respectively. As a result, the routing conceptualization is a modification of this matrix, where each column is shifted to induce a temporal delay.

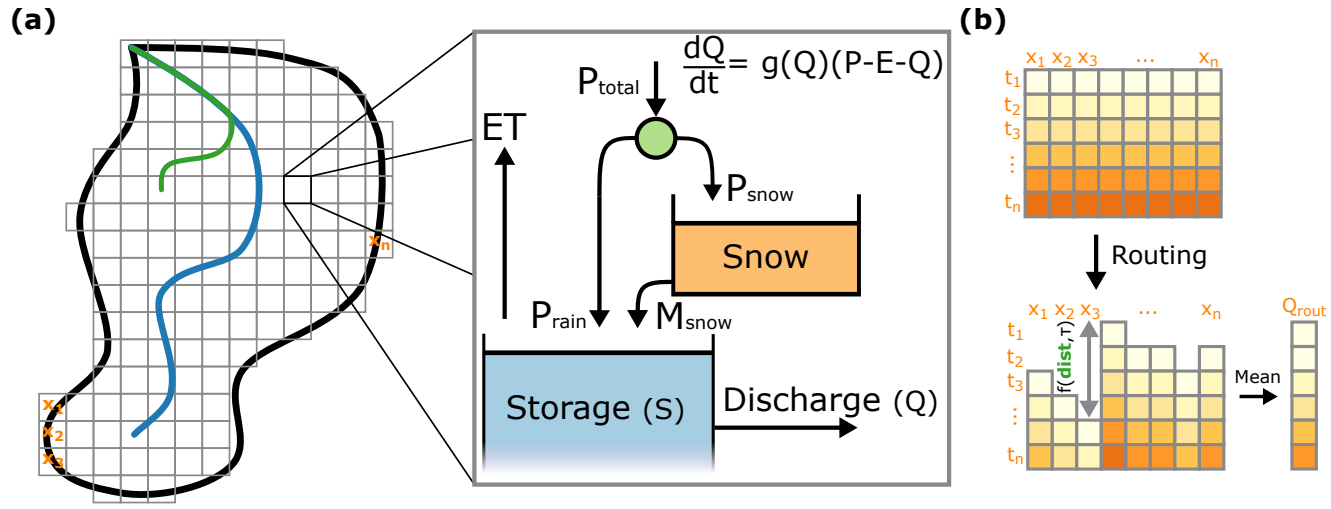

Figure 1. Efficient distributed implementation of the simple dynamical systems approach, which is solved for each pixel. The left hand side of panel a represents a catchment, with a regular grid covering the catchment area. The green line indicates the flow path on which the routing lag is based (green "dist" in panel b). Panel b shows how the catchment is translated to a matrix to allow for computationally efficient calculations, and how the matrix is modified in the routing algorithm, by shifting each column based on the distance to the outlet and the routing parameter τ.

2.2 Discharge sensitivity

As previously mentioned, the sensitivity function is required to translate changes in storage to changes in discharge. ~~This function can have any shape.~~ Kirchner (2009) originally presented a simple ~~power-law~~ power–law version of this sensitivity function for purposes of illustration:

$$g(Q) = \frac{\mathrm{d}Q}{\mathrm{d}S} = aQ^b, \tag{4}$$

where a and b define the slope and ~~intersect~~ intercept of the sensitivity function in ~~log-log~~ log–log space. This ~~power-law~~ power–law relation has been widely used in experimental and theoretical studies (e.g. Troch et al., 1993; Brutsaert and Lopez, 1998; Tague and Grant, 2004; Rupp and Selker, 2006; Lyon and Troch, 2007; Rupp and Woods, 2008). This relatively simple sensitivity function allows the translation of discharge into storage, using the following equation:

$$\int \mathrm{d}S = \int \frac{\mathrm{d}Q}{g(Q)}, \tag{5}$$

$$S(Q) = \begin{cases} \frac{1}{a}\frac{1}{1-b}Q^{1-b} + S_0, & b \neq 1 \\ \frac{1}{a}\ln(Q) + S_0, & b = 1, \end{cases} \tag{6}$$

where S_0 is the integration constant, meaning that only relative storage changes can be obtained using this method. The value of b affects the meaning of S_0, as described by Kirchner (2009). However, most catchments show recession behaviour that

differs from a ~~power-law~~ power–law relation between $\mathrm{d}Q/\mathrm{d}S$ and Q (Kirchner, 2009; Teuling et al., 2010; Krier et al., 2012; Adamovic et al., 2015). Therefore, a more complex formulation of the sensitivity function was also proposed by Kirchner (2009):

$$\ln(g(Q)) = c_1 + c_2 \ln(Q) + c_3 (\ln(Q))^2, \tag{7}$$

where c_1, c_2 and c_3 are the three parameters of this quadratic equation. This quadratic equation allows for a concave relation between the recession rate and the discharge in ~~log-log~~ log–log space. This equation has one downside, however: since it is shaped like a parabola in ~~log-log~~ log–log space, there is always an optimum in the discharge sensitivity. This implies that with an increasing Q, the system becomes less sensitive at some point. This behaviour is unrealistic and unwanted when performing automatic calibration or random parameter sampling runs. Therefore, we have added an additional term to the original ~~power-law~~ power–law equation, which accounts for the concave shape of the sensitivity function:

$$g(Q) = aQ^b \cdot \mathrm{e}^{\gamma/Q}, \tag{8}$$

where a, b and γ are the three parameters describing the shape of the discharge sensitivity. However, we have rewritten this equation to include all parameters in the exponent term, as this improves computational efficiency:

$$g(Q) = \mathrm{e}^{\alpha+\beta\cdot\ln(Q)+\gamma/Q}, \tag{9}$$

with $\alpha = \ln(a)$ and $\beta = b$ from the Eq. (8). In Fig. 2, the effect of each parameter on the shape of the sensitivity function is presented. Discharge observations from Teuling et al. (2010) are also included to indicate the importance of the γ parameter. This equation has the benefit that it can be rewritten to the original power law equation, if $\gamma = 0$:

$$g(Q) = \mathrm{e}^{\alpha+\beta\cdot\ln(Q)} = \mathrm{e}^{\alpha} \cdot Q^{\beta}, \tag{10}$$

where $a = \mathrm{e}^{\alpha}$ and $b = \beta$ from Eq. (4). This concept allows us to use parameters from previous studies, and to transform the discharge time series to a storage time series according to Eq. (5). A disadvantage of this new sensitivity function is that, due to the addition of the γ/Q term, there is no longer an analytical solution, but the equation can still be integrated numerically.

2.3 Snow processes

In our distributed implementation, we allow precipitation to fall as snow. Snow is treated as a separate storage (see Fig. 1a), where snowmelt is added to the simple dynamical systems approach in the form of liquid precipitation, following the methodology of Teuling et al. (2010). We assume snowmelt to be dependent on both temperature and radiation, following the restricted ~~degree-day~~ degree–day radiation balance approach (Kustas et al., 1994). The snow storage is conceptualized based

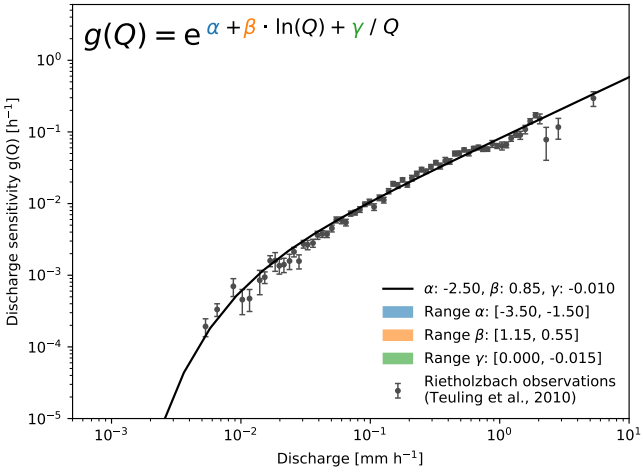

Figure 2. ~~Parameterization~~ Parametrization of the discharge sensitivity, including the effects of the three parameters. α affects the intercept, β affects the slope and γ affects the curvature at low discharge values. Parameter values for each parameter are given in the legend as their ~~min-max~~ min–max range.

on the following equations:

$$\frac{\mathrm{d}S_{\mathrm{snow}}}{\mathrm{d}t} = P_{\mathrm{snow}} - M_{\mathrm{snow}}, \tag{11}$$

$$P_{\mathrm{snow}} = \begin{cases} P_{\mathrm{total}} & \text{if } T <= T_0 \\ 0 & \text{if } T > T_0, \end{cases} \tag{12}$$

$$M_{\mathrm{snow}} = \begin{cases} ddf \cdot (T - T_0) + rdf \cdot R_{\mathrm{g}} & \text{if } M_{\mathrm{snow}} \cdot \Delta t <= S_{\mathrm{snow}} \\ \dfrac{S_{\mathrm{snow}}}{\Delta t} & \text{if } M_{\mathrm{snow}} \cdot \Delta t > S_{\mathrm{snow}}, \end{cases} \tag{13}$$

where S_{snow} is the total snow storage in mm, P_{snow} the precipitation falling as snow in mm h^{-1}, M_{snow} is the snowmelt in mm h^{-1}, T is the air temperature in °C, T_0 is the critical temperature for snowmelt in °C, ddf is the degree-day factor in mm h^{-1} °C^{-1}, rdf is the conversion factor for energy flux density to snowmelt depth in mm h^{-1} (W m^{-2})$^{-1}$, R_{g} is the global radiation in W m^{-2}, and Δt is the time step in hours. If no radiation observations are available, the snowmelt equation is modified to the normal degree-day method.

In Fig. 3, the effect of this snow conceptualization is presented using synthetic forcing data. In this figure, one can see that radiation can cause snow to melt even when temperatures are still below the critical temperature (0°C in this example). If the temperature exceeds this threshold, radiation amplifies the melting of snow, resulting in an earlier depletion of the snow storage. Finally, if snow processes are not relevant in the region of interest, the snow conceptualization can be turned off to further reduce the computational demand (see the dashed line in Fig. 3 for the resulting discharge simulation).

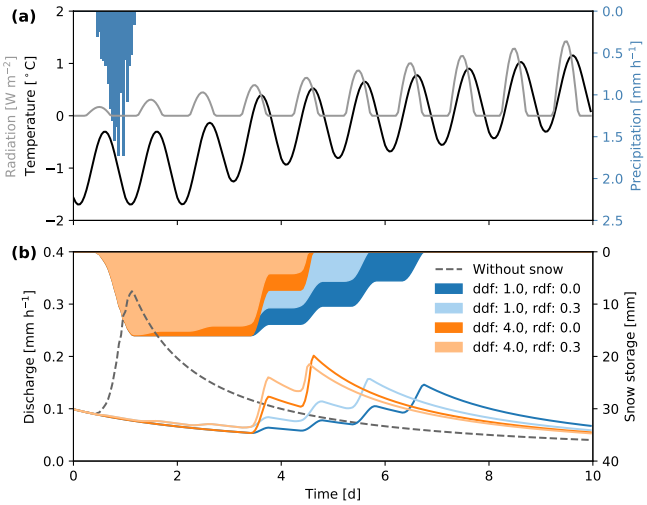

Figure 3. Simulation results with and without the snow conceptualization. Top panel shows the model input, bottom panel shows the model output. The four hydrographs represent runs with different snow parameter values (see legend, in mm h^{-1}). We fixed the critical temperature T_0 to 0°C. To visualize the influence of the different parameters, we used unrealistically high ddf and rdf values.

2.4 Flow routing

A simple efficient routing conceptualization was added to the distributed model to transport water from each grid cell to the outlet of the catchment. In most studies applying the simple dynamical systems approach, a constant delay factor was added to the generated runoff time series, in order to account for the delay caused by the river network. Since our distributed implementation has pixels at ~~distinct~~ distinctly different locations, we need to account for attenuation caused by the river network. The routing concept presented here is based on the ~~width function~~ already existing width function concept (Kirkby, 1976), where ~~we assume~~ it is assumed that the stream network ~~applies~~ induces a temporal delay proportional to the distance to the outlet. This ~~distance~~ can be seen as a width function based unit hydrograph without diffusion, and goes back at least 20 years (e.g. Franchini and O'Connell, 1996). The distance to the outlet is combined with the travel speed parameter (τ) to determine the lag time for each pixel (see the green line and shifting of columns in Fig. 1). An example of this concept is presented in Fig. 4a.

The width function in Fig. 4a reflects the shape and stream network of a synthetic catchment. The distance of each pixel can be translated to a time delay using the travel speed parameter τ. This delay is in discrete steps (see Fig. 4b), determined by the time step of the model. ~~Lower flow speed values~~ Slower flow speeds result in larger time delays, and vice versa. The hydrograph in Fig. 4c shows that peaks are more attenuated with ~~lower flow speed values~~slower flow speeds. In this example, we assume homogeneous precipitation across the entire catchment, but in reality, the heterogeneity of precipitation events also influences how the discharge peak is attenuated. This concept does not include diffusion of ~~a flood wave~~flood waves, but only incorporates advection.

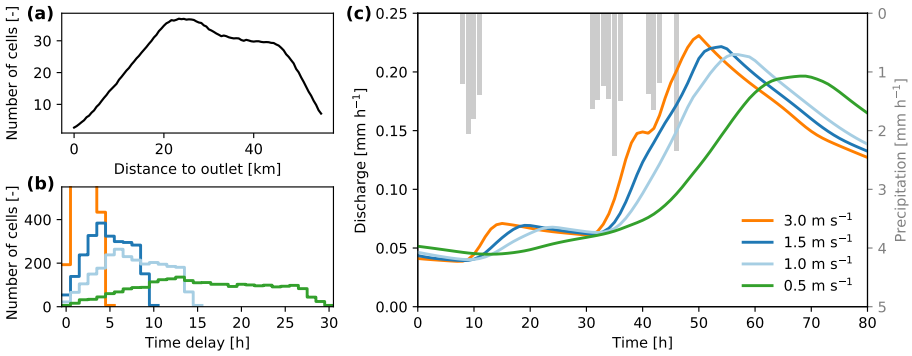

Figure 4. The routing concept visualized: a) the width function of a synthetic catchment, b) the corresponding travel speed given four different τ values, and c) the effect of different τ values on the hydrograph.

3 Technical aspects

3.1 Model implementation

One of the main aims of this model was computational efficiency. ~~Low-level languages generally perform faster calculations, but this comes at the price of user-friendliness and ease-of-use. The modelimplementation is focussed on efficiency, but~~ However, we do not claim that we have built the most computationally efficient conceptual hydrological model, since we sacrificed some calculation time ~~is sacrificed for a higher-level but more user friendly language.~~ for the user-friendly and hugely popular Python programming language. By choosing this language, we encourage users of the model to adapt, improve and change all components of the model, in order to find answers to their research questions. The model is written in Python 3.6 and largely ~~utilizes~~ uses the Numpy library. ~~The efficient implementation of arrays in Numpy allows for~~ Numpy uses C libraries to ensure fast computations over entire arrays, something the base functions of Python do not offer. Due to the simplicity of the simple dynamical systems approach, we need to solve only one equation (besides the snow conceptualization). Numpy allows functions to be vectorized: receiving and outputting an array of values, while applying the same function ~~on~~ to each individual value. This is computationally more efficient than the ~~step-by-step~~ step–by–step application of the same function to each element in the array.

To get an idea of the computational efficiency of the model, example run times are presented in Fig. 5. We ran the model for 3 months at an hourly time step using synthetic forcing data for a wide range of model cells, reflecting different catchment sizes or model resolutions, to get an idea how the computational demand scales with catchment size or resolution. We separated the time spent in the numerical solver, the IO operations, and the routing module. In the IO operations, we include reading the settings file, reading the input data, and writing the model results. The grey bars in Fig. 5 represent the "traditional" modelling approach, where the numerical solver is called for each individual pixel and time step. This extra ~~for-loop~~ for–loop drastically increases runtimes by a factor of more than 10 (this is also a property of Python, being an interpreted language instead of a compiled language). ~~Simulating~~ To demonstrate the enormous potential of this model to explore uncertainty and spatial

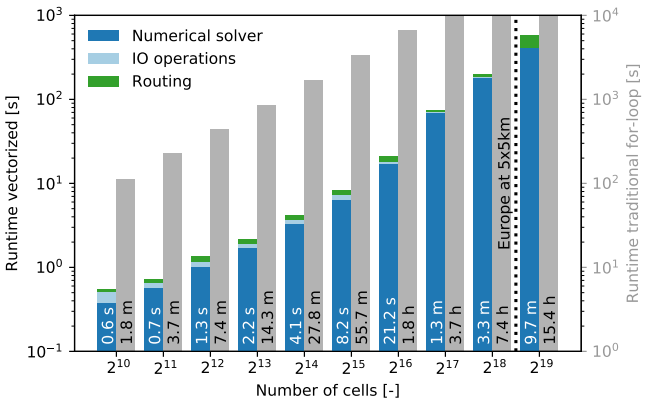

Figure 5. Runtimes for a simulation of three months with an hourly time step with varying number of pixels, ran on a single core of a normal desktop (Intel Core i7-6700, 16GB RAM). The grey bars represent model run times where the numerical solver is called for each individual pixel, and are keyed to the right hand scale, making the difference in runtimes appear smaller than it really is. The dotted line represents the number of pixels when simulating Europe at 5×5 km^2 resolution.

patterns in parameters: simulating Europe for three months at hourly time steps and at a resolution of 5×5 km^2 (roughly $2^{18.6}$ pixels) would take approximately 5 minutes with the efficient dS2 model~~, demonstrating the enormous potential of this model to explore uncertainty and spatial patterns in parameters~~. The runtimes show a small inconsistency in the increase of runtimes with the increase in pixels. This is most likely the result of an internal Python or Numpy threshold. The current version of dS2 does not yet support automatic multi-threading, which is expected to even further reduce runtimes, especially in large basins.

When applying the model to very large basins and/or running for very long periods, the random access memory (RAM) can become a limiting factor, as storing data in the RAM is by far most efficient. To prevent exceeding the RAM, the user can define the maximum amount of RAM the model is allowed to use, and dS2 will chunk the input and output data accordingly. For example, running the model for a basin with 1500 pixels for a period of 4 years on a hourly time step would require approximately 800MB. Additionally, to reduce the time spent with reading and writing the files on the disk, we rely on a special data format: Numpy memmap. This data file directly maps arrays to the hard drive without storing any metadata, to ensure fast reading and writing. The downside is that this format does not have any metadata, meaning the shape and data type needs to be known prior to reading the file. In order to store the model output in a more common format, the model can transform the Numpy memmap data to the more commonly used NetCDF format, including ~~meta-data~~meta–data.

3.2 Adaptive time stepping

~~Since this concept~~ When solving a differential equation, one would ideally opt for an analytical solution. However, many equations (including Eq. 3) do not have an analytical solution, so one needs to use numerical solvers to solve the differential equation. The downside of numerical methods is that they might introduce numerical errors and/or numerical instability,

as they are approximations of the analytical solution. Many different numerical methods exist, varying in complexity and numerical accuracy. The importance and potential problems of numerical solutions have been described and studied extensively (Clark and Kavetski, 2010; Kavetski and Clark, 2010, 2011). For example, the most simple method is the first order explicit Euler scheme, meaning that the value at t_{n+1} is based on the values given at t_n. This method is the simplest approximation of the value at t_{n+1}, but is as a first order method also prone to introduce large numerical errors. A more robust and popular method is the explicit Runge–Kutta 4 scheme, which uses four estimations (fourth order) between t_t and t_{n+1} to give a more accurate estimation of the value at t_{n+1}. Furthermore, using higher order (or implicit) solvers reduces the risk of numerical instability. Many more methods are available, which are higher order and/or use different ways to calculate the "intermediate" steps. Numerical solvers in most hydrological models are rather basic, although it is known that the numerical solver can significantly influence the model output (Schoups et al., 2010). In particular in strongly non-linear problems, a robust numerical solver is important.

Since the simple dynamical systems approach allows for non-linear reservoirs, the simulated response can vary over several orders of magnitude within a single time step. This non-linearity can cause numerical errors, which can be prevented by reducing the time step. Several options are available, where the ~~Cash-Karp~~ Cash–Karp method is the typical textbook approach to explicitly solving differential equations (Cash and Karp, 1990). This method is based on the ~~Runge-Kutta~~ Runge–Kutta scheme, and uses the difference between fourth and fifth order ~~estimations~~ estimates to measure the potential numerical error. This can be used to reduce the time step if the difference exceeds a certain threshold. However, to ensure optimal computational efficiency, we decided to incorporate the knowledge about the differential equation (Eq. (3)) to determine whether time step reduction is necessary. ~~We~~ Therefore, we have implemented the explicit fourth order ~~Runge-Kutta~~ Runge–Kutta (RK4) scheme ~~,~~with an adaptive time stepping scheme. We have identified cases where the RK4 scheme can become numerically unstable, and used these cases to reduce the internal time step. These cases are explained in more detail below. This adaptive time stepping scheme is visualized in Fig. 6.

The adaptive time stepping scheme operates in three steps. In the first step, the discharge for the next time step is calculated using the explicit ~~Runge-Kutta~~ Runge–Kutta 4 scheme. For both time steps, the discharge sensitivities are calculated based on the following equation:

$$gQ_{\mathrm{diff}} = \frac{|g(Q)_{t2} - g(Q)_{t1}|}{\min(g(Q)_{t2} - g(Q)_{t1})}, \tag{14}$$

where the numerator represents the absolute difference between the two discharge sensitivities, and the denominator represents the lowest discharge sensitivity to ensure the comparison also works during the falling limb of the hydrograph. The time step will be reduced whenever at least one of two conditions ~~are~~ is met: (1) when $g(Q_{t2}) \cdot \Delta t > 1$, or (2) when gQ_{diff} exceeds a certain threshold (gQ_{mdiff}). The first condition requires a smaller time step, since values of $g(Q) \cdot \Delta t > 1$ indicate that the system is extremely sensitive, and that a smaller time step is required to optimally account for this sensitivity. For this condition, the time step is reduced according to the following equation:

$$\frac{\Delta t}{\Delta t_{\mathrm{new}}} = \max(5, \min(g(Q_{t2}) \cdot 10, 50)), \tag{15}$$

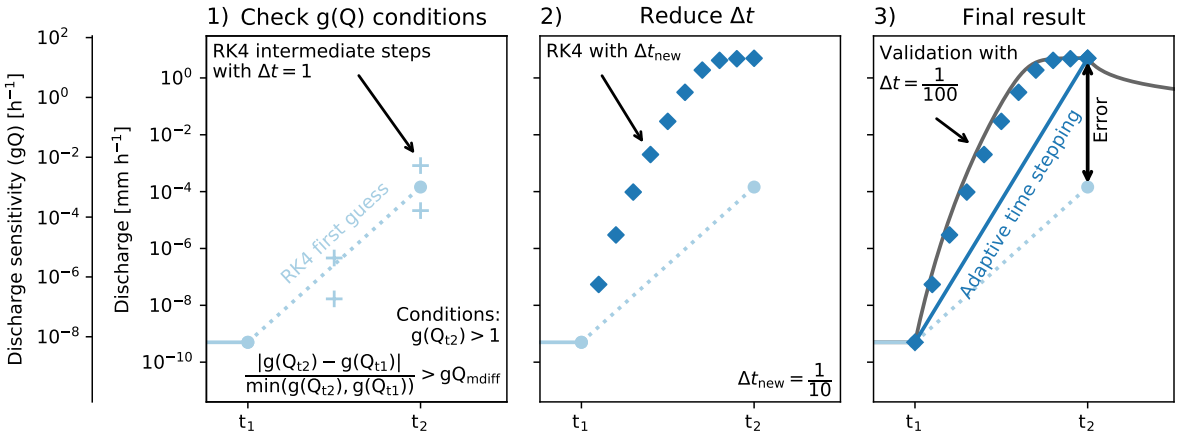

Figure 6. Three steps in the adaptive time stepping scheme. Note that in extreme cases, both Q and $g(Q)$ can change over several orders of magnitude between t_1 and t_2, as can be seen from both y-axes. The plus symbols in step 1 indicate the intermediate values by the RK4 solver, which combine into the blue circle at t_2. The blue diamonds in step 2 show the intermediate steps calculated with RK4 but at a smaller timestep. Only the value at t_2 is stored. The error in step 3 represents the numerical error without the adaptive time stepping scheme.

where the reduction in Δt is dependent on the size of $g(Q_{t2})$. We defined a lower and upper limit to the time step reduction, to ensure that the solver gains enough precision but does not spend too much time on a single time step. Both ~~values~~ upper and lower limits can be changed by the user. The second condition is triggered when $g(Q)$ covers several orders of magnitude in a single time step. A threshold gQ_{diff} is defined, describing how many times $g(Q_{t2})$ is allowed to deviate from $g(Q_{t1})$. If this threshold is exceeded ($gQ_{\text{diff}} > gQ_{\text{mdiff}}$), a smaller time step is required since RK4 is not able to accurately solve the differential equation over this many orders of magnitude. The resulting reduced time step is based on the following equation:

$$\frac{\Delta t}{\Delta t_{\text{new}}} = \max(5, \min(gQ_{\text{diff}}^{\sout{dt\_reduction}\,\text{dt\_reduction}}, 50)), \tag{16}$$

where the reduction in Δt is dependent on the difference between $g(Q_{t1})$ and $g(Q_{t2})$, raised to the power ~~*dt_reduction*~~dt_reduction, and the same lower and upper ~~limit~~ limits to the time step reduction are used. The RK4 scheme is used over these reduced time steps, until the original time step (t_2) is reached. This final value is saved as output, and the intermediate steps are discarded. In Fig. 6 step 3, we validate the adaptive time stepping scheme with an even finer time step ($\frac{1}{100}$) which we assume ~~to approach~~ approaches the analytical solution, and we conclude that the adaptive time stepping scheme yields reliable results. This figure shows that the adaptive time stepping scheme avoided a large numerical error (the difference between the two solutions: $Q_{t2} \approx 10^0$ mm h^{-1} vs $Q_{t2} \approx 10^{-4}$ mm h^{-1}).

During periods with low discharge and relatively high evaporation, the numerical solver can result in negative discharge amounts. To prevent this from happening, we define a threshold for discharge (which thus corresponds to a threshold storage level), below which no evaporation is allowed to occur. This mimics evaporation reduction during periods with low storage volumes, as discharge and storage are directly linked. For the pixels where the discharge is below this threshold, the evaporation

rate is set to 0 mm h^{-1}. The value of the discharge threshold is currently set at 10^{-4} mm h^{-1}, but can be altered by the user. All model parameters are presented in Table A1.

3.3 Closure of the water balance

The concept of this model is based on the water balance, yet it does not explicitly solve the water balance as most hydrological models do. The water balance is indirectly solved by calculating changes in storage. To check whether the model still respects the water balance, the closure of the water balance is calculated using the following equation:

$$wbal\_error\phi = P_t - E_t - \int_{t-1}^{t} Q - \Delta S, \tag{17}$$

$$\Delta S = S(Q_t) - S(Q_{t-1}), \tag{18}$$

where ~~*wbal_error*~~ ϕ is the error in the water balance for each time step t, which ideally should result in a value of zero. The change in storage can be calculated using the storage at the beginning and at the end of the time step. The integral over Q indicates that one needs to consider the volume of water that is discharged during the time step. This depends on how the variables are considered in time within the model. Figure 7a explains how the different fluxes are positioned within the model. The model assumes that the precipitation and evaporation values are summations of the entire duration of a single time step. The resulting discharge values are, however, only representative at the end of the time step, and not a summation over the time step. Due to the strong non-linearity of the system, the total volume of water discharged during a single time step cannot be represented by the discharge at the end of the time step, as indicated by the dotted red line in Fig. 7a.

To estimate the volume of water discharged during the time step, one needs a discharge value that is representative for the volume of water discharged during that time step. The easiest way to calculate this ~~, it~~ is to take the average of the discharges at time ~~step~~ steps t and $t-1$. However, if the sensitivity function is strongly non-linear, ~~the average discharge~~ this average might not be representative for the volume of water discharged during that time step. To improve the volume estimation, one can subdivide each time step, and base the volume estimation on the mean discharge of the resulting shorter steps ($Q_{internal}$). This comparison is shown in Fig 7b. The ~~gray~~ grey bars represent the error in the water balance when the total discharge volume is estimated using the discharge averaged over time steps t and $t-1$. This error does not reduce with an increase in internal time steps, as the discharge prediction at t changes only marginally. However, if the total discharge volume is estimated using the discharge at each internal time step, the total error in the water balance is reduced from $10^{-2}\%$ to $10^{-8}\%$. This indicates that the model concept is able to successfully close the water balance, given that the discharge volume per time step is accurately accounted for. Please note that this example is to show that the concept successfully closes the water balance ~~, but the~~ at each time step. The current version of the model only outputs discharge at the end of the time step, as calculating $Q_{internal}$ does not add to the numerical accuracy of the discharge calculation at the end of the timestep.

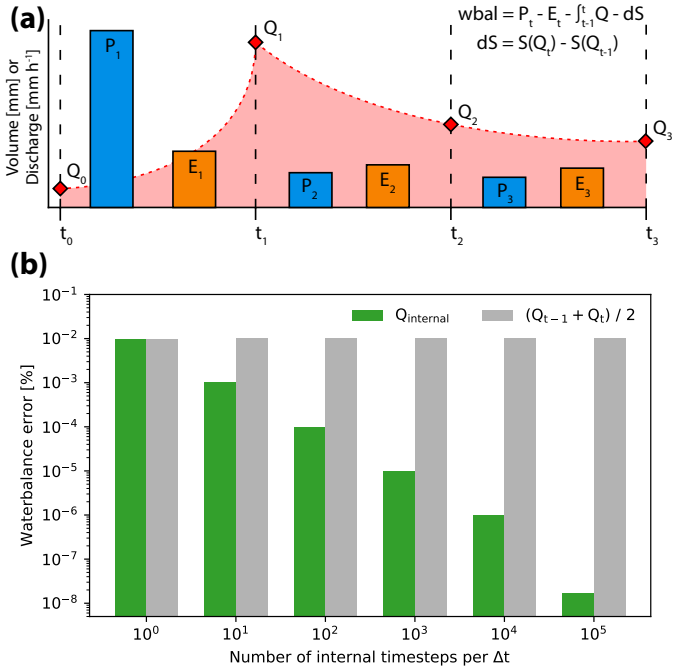

Figure 7. Closure of the water balance per time step. Panel (a) shows the timing of the different variables. Red area under the dotted line indicates the total volume of water discharged per time step. Panel (b) shows the average absolute error in the water balance ~~per timestep~~ when the volume of water is estimated with ~~different~~ smaller internal time steps ($Q_{internal}$), where the error is defined as the percentage of the total precipitation during the simulation period.

4 Parameter sensitivity

We investigated the response of the model to ~~each parameter~~ the five main parameters (α, β, γ, ϵ, τ; excluding the snow parameters) by plotting the response surface over realistic parameter ranges. These ranges are based on comparing the shape of the resulting discharge sensitivity function with the previous studies that use the SDS approach. The results of this analysis are presented in Fig. 8. We selected the ~~Kling-Gupta~~ Kling–Gupta efficiency (KGE) as the performance metric. We created a synthetic time series of observations ~~that has~~ based on parameters that are in the middle of each subplot, which are used to calculate the KGE. It is striking that the three ~~$g(Q)$ parameters~~ discharge sensitivity parameters (α, β and γ) show large regions with similar model performance (the dark blue regions), and seem to be correlated, which Melsen et al. (2014) also showed for the α and β parameters for a lumped application. However, based on ~~the theoretical considerations (Troch et al., 1993; Brutsaert and Lopez, 1998; Tague and Grant, 2004; Rupp and Selker, 2006; Lyon and Troch, 2007; Rupp and Woods,~~ theoretical considerations (Troch et al., 1993; Brutsaert and Lopez, 1998; Tague and Grant, 2004; Rupp and Selker, 2006; Lyon and Troch, we can conclude that at least the α and β parameters are required to optimally capture the discharge sensitivity, and based on other studies using the simple dynamical systems approach ~~(Kirchner, 2009; Teuling et al., 2010; Krier et al., 2012; Adamovic et al., 2015)~~

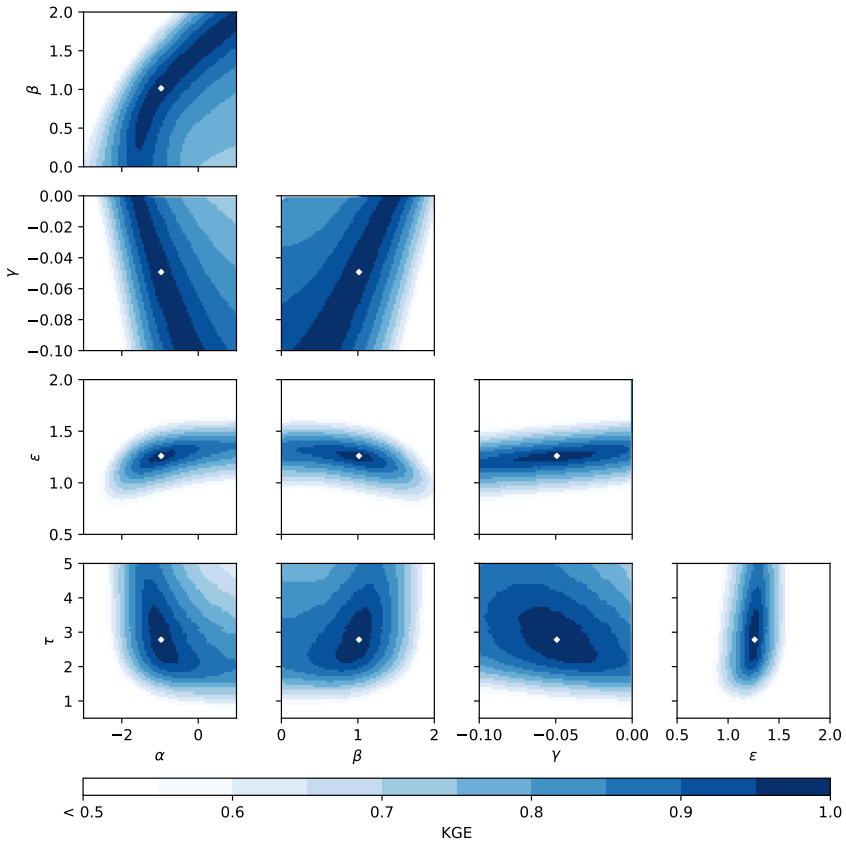

Figure 8. Response surface plots for all main parameter combinations with the ~~Kling-Gupta~~ Kling–Gupta efficiency (KGE) as the performance metric for a synthetic experiment. The white dot in the middle of each graph represents the location where KGE is equal to 1. Each plot consists of 4900 model runs ~~,~~ where the model has 200 cells, ~~ran~~ and was run for one year on hourly timestep ~~,~~ and took 1 hour and 15 minutes to run on a normal desktop (Intel Core i7-6700, 16GB RAM).

(Kirchner, 2009; Teuling et al., 2010; Krier et al., 2012; Adamovic et al., 2015), that a third parameter (γ in this case) is required to capture the curvature in the $g(Q)$ relation. These response surface graphs indicate that local optimization algorithms might struggle to find the global maximum due to the large equifinality regions. We therefore recommend not to use a single parameter combination, but rather to use multiple parameter sets to account for equifinality.

Furthermore, we also analysed the parameter sensitivity according to Sobol′ (2001), Saltelli (2002), and Saltelli et al. (2010). The global sensitivity analysis method is designed to analyse the sensitivity ~~to~~ of different performance statistics to each parameter. Sobol' sensitivity analysis ~~creates a~~ works optimally in a case where parameters are not correlated. Even though this condition is not fully met in our model application, we use Sobol' because it is the most widely used method to investigate parameter sensitivity. Furthermore, the aim of this sensitivity analysis is to give a first impression of the influence of the parameters on performance metrics. To perform this analysis, a set of $n \cdot (2k + 2)$ ~~combinations~~parameter combinations is

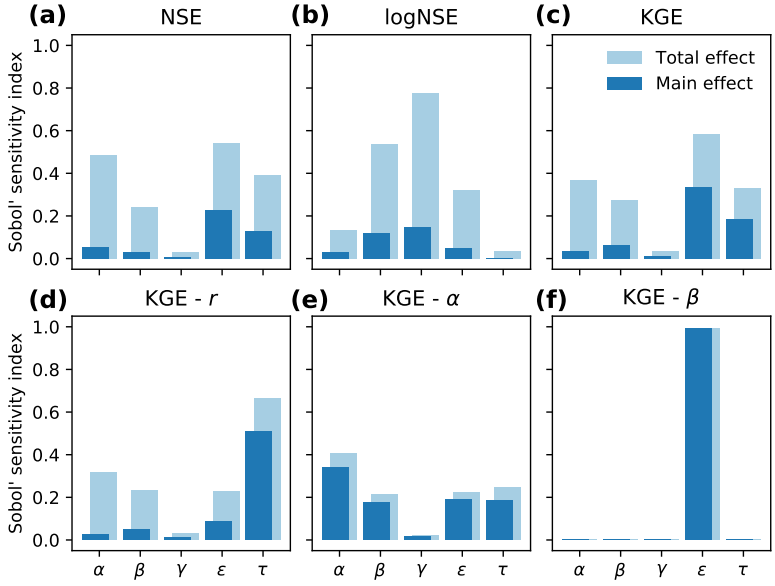

Figure 9. Sobol' parameter sensitivity for a synthetic experiment, shown for different performance statistics. The main effect presents the sensitivity induced by that parameter alone, and the total effect quantifies the sensitivity when parameter interactions are taken into account as well. The bottom row (panels d to f) ~~show~~ shows the three components of the ~~Kling-Gupta efficiency~~Kling–Gupta Efficiency: Pearson correlation coefficient (KGE - r), ratio of variability (KGE - α) and bias (KGE - β).

required, where k is the number of parameters, and n is the number of samples that sample the parameter space. ~~For this study, we~~ We chose a sample of $1,000$, and focused on the five main parameters (α, β, γ, ϵ, τ, excluding the snow parameters) resulting in $12,000$ parameter sets. Usually, this is a rather computationally expensive method, but due to the computational efficiency of this model, we were able to run all parameter sets in just under 6 hours (model with 100 cells, simulation period of 2 years on hourly time step, on a normal desktop with an Intel Core i7-6700 and 16GB RAM). The parameter boundaries were set to the same values as used in the response surface plots. For each parameter combination, multiple performance statistics were calculated. Next, Sobol' sensitivity analysis is able to extract the influence of each parameter on the variation in the performance statistic caused only by that parameter ("main effect"), and the influence of the parameter including all variance caused by interactions with other parameters ("total effect") (Sobol′, 2001; Saltelli, 2002; Saltelli et al., 2010). The results from the analysis are presented in Fig. 9.

It is clearly visible that parameter sensitivity depends on the performance metric used. The ~~Nash-Sutcliffe and Kling-Gupta efficiencies~~ Nash–Sutcliffe Efficiency (NSE) and Kling–Gupta Efficiency (KGE) show roughly the same response, as is expected. ~~Nash-Sutcliffe~~ Nash–Sutcliffe calculated on the logarithmic discharge values shows a very different response, with γ being one of the most sensitive parameters. This is in line with our expectations, as the γ parameter describes the downward curvature of the $g(Q)$ function, where it mostly affects the lower discharge volumes. ~~As a result~~Additionally, this parameter is not very prominently visible in the ~~NS~~ NSE and KGE sensitivity plots, since these metrics tend to put more focus on higher

discharges. These graphs also show that there are some parameter ~~correlations~~ interactions influencing the results, indicated by the difference between the main and total effects. This was already visible in Fig. 8, where we see correlations between the three $g(Q)$ parameters. Since this analysis is performed to give a first evaluation of how the parameters affect model output, we will focus on the main effect.

The most interesting patterns are visible in the bottom three subplots, where the three components of the KGE are presented. The correlation coefficient (KGE - r, Fig. 9d) is most sensitive for the routing parameter τ, since this parameter essentially deals with the timing of discharge peaks. The ratio of variability (KGE - α, Fig. 9e) does not show ~~a single~~ any individual parameter to be most important, yet all parameters show some degree of sensitivity. For the bias (KGE - β, Fig. 9f), the evaporation parameter ϵ is by far the most sensitive parameter, since it determines the total volume of water that evaporates and thus the total volume of discharge. All other parameters can only control either the response of discharge to precipitation or the timing of the peaks. Since the roles of the parameters ϵ and τ are relatively clearly defined, predominantly affecting the bias and correlation, respectively, we hypothesize that ~~this information~~ these roles can be used for more efficient optimization procedures by reducing the number of dimensions. One could optimize the ϵ parameter only on the bias of the simulations, followed by an optimization of the τ parameter on the correlation. Finally, the three $g(Q)$ parameters can be optimized on either the ratio of variability or the total ~~Kling-Gupta~~ Kling–Gupta efficiency. This reduces the optimization from a 5-dimensional ~~space~~ problem to two 1-dimensional problems and a single 3-dimensional problem.

5 Example application

In this section, we apply the model to the Thur, a mesoscale basin (1700 km^2) in the Swiss Alps. This basin was selected since it contains many measurement locations, where the catchment used by Teuling et al. (2010) is one of the sub-basins. Since Teuling et al. (2010) showed that the SDS concept is able to simulate the discharge at small scale, we take this opportunity to see how the model performs at different spatial extents. The model was applied at a 1 km^2 resolution, using distributed forcing. We used the same forcing data as Melsen et al. (2016a), where the data is interpolated using the pre-processing tool WINMET of the PREVAH modelling system (Viviroli et al., 2009; Fundel and Zappa, 2011). For a more detailed explanation of the basin and the data used, we refer to Melsen et al. (2016a).

For this application, we focus on three sub-basins in the Thur to validate the model at three different spatial extents (see Fig 10a). We used a Monte Carlo approach to generate $25,000$ parameter sets ~~, with parameter boundaries~~ to ensure good coverage in the parameter space. The parameter boundaries are based on the ranges ~~presented~~ used in Fig. 8. We based the snow parameters on the study by Teuling et al. (2010). To capture the spread in parameters as a result of equifinality, the 100 best runs based on the ~~Kling-Gupta efficiency~~ Kling–Gupta Efficiency (KGE) are selected for each sub-basin. The resulting discharge time series are presented in Fig. 10b to Fig. 10d.

In Fig. 10b to Fig. 10d the simulated discharge is compared with the observed discharge for three different basins. The three basins are ordered from small to large. The simple dynamical systems approach has already been applied to the Rietholzbach, the smallest of the three basins in Fig. 10, by Teuling et al. (2010), but in a lumped fashion. We see that the distributed version

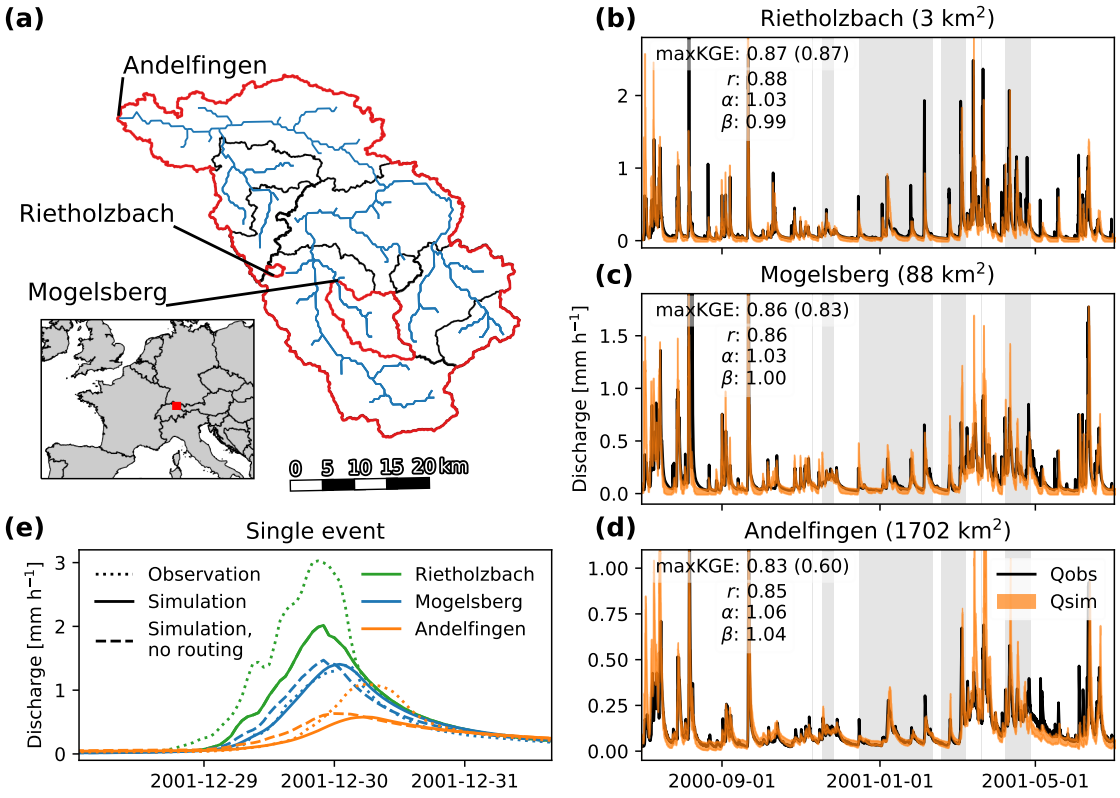

Figure 10. Application of dS2 to the Thur basin in Switzerland~~, where~~. Panel (a) depicts the stream network and ~~sub-basins~~ the catchments used in this example are highlighted in red: Rietholzbach, Mogelsberg and Andelfingen. ~~Graphs~~ Panels (b) to (d) present the model ~~performance~~ output in the three basins~~across~~, covering different orders of ~~sizes~~catchment size. The shaded orange region shows the minimum and maximum discharge of the ~~selected~~ 100 best runs out of 25,000, selected based on KGE value. Periods where more than 20% of the entire basin is covered with snow are indicated with the grey background. Performance values show the statistics for the best model run, with the value between brackets showing the model performance when the routing module is disabled. Graph (e) shows how a single precipitation event is translated through the different (sub-)basins, where the ~~light colours indicate~~ dotted line indicates the observed discharge, the ~~dark lines represent~~ solid line represents the simulated discharge, and the dashed ~~lines represent~~ line represents the simulated discharge without a temporal delay induced by the routing module. Note that the simulation result without routing for Rietholzbach overlaps the simulation result with routing.

of this concept is able to correctly simulate the discharge in this basin, but also in the bigger basins. In the top left corner of each subplot, 4 performance statistics are presented based on the run with the highest ~~Kling-Gupta efficiency: the Kling-Gupta efficiency~~ Kling–Gupta Efficiency: the Kling–Gupta Efficiency (maxKGE), Pearson correlation coefficient (r), ratio of variability (α), and bias (β). The KGE value within brackets represents the ~~Kling-Gupta efficiency~~ Kling–Gupta Efficiency without the routing module. These metrics show that there is no substantial decrease in model performance with increasing catchment size. This indicates that the model is able to correctly simulate the discharge, independent of the catchment size. We do see, however, a clear decrease in the KGE values when no temporal delay is added via the routing scheme. In the smallest basin, both KGE values show the same result, yet in the largest basin, we see a substantial decrease in model performance. This indicates, in line with expectations, that the routing module is required to optimally simulate larger basins.

To elaborate this, we zoom in to a single event in Fig. 10e. This graph shows the discharge response of the three sub-basins, and the effect of the routing module. For this event, we selected the one parameter set from the 100 sets that performed best ~~for this specific event~~over the period depicted in panels (b) to (d). We see that, although the magnitude of the peak is not fully captured in the Rietholzbach ~~and Andelfingen basins~~basin, the timing of the peak is well simulated~~in all three basins . The run~~ . Since this is a very small basin, the time lag induced by the routing module is less than one hour, meaning that the run without the routing module gave the same result. At Mogelsberg, both the timing and magnitude of the discharge peak are well simulated by dS2. The routing module adds a correct temporal delay to the discharge peak. At the main catchment outlet (Andelfingen), we again see that the model correctly simulates the timing of the discharge, but struggles to reach the correct magnitude. This is likely related to errors in the precipitation product rather than the model structure, as dS2 struggled to reach good performance in two basins in the north western part of the catchment (Frauenfeld and Wängi). This, together with the too low discharge peak in the Rietholzbach basin, might explain the lack of discharge passing Andelfingen. Similar to Mogelsberg, the routing module does add a correct filtering of the discharge signal due to the distribution of the stream network. Overall, the runs without the routing module ~~indicates~~ indicate that the impact of the routing module depends on the size of the catchment, where the difference between these runs is largest in the biggest catchment. The differences between these runs explain the reduction in model performance in Fig. 10b to Fig. 10d, measured using the KGE without routing.

6 Discussion

As stated in the introduction, the aim of this paper is not to present the best or most comprehensive conceptual ~~rainfall-runoff~~ rainfall–runoff model, but to develop a model that can more easily be used to perform studies that require a large number of runs, such as uncertainty or sensitivity studies. As a result, this model concept has several limitations, either linked to the original simple dynamical systems approach or linked to the (distributed) implementation of this concept.

A limitation of the simple dynamical systems approach is that it assumes that the ~~storage-discharge~~ storage–discharge relation is the same ~~in~~ on the rising limb and ~~in~~ on the falling limb of the hydrograph. Studies have shown that hysteresis exists in multiple basins, especially those dominated by a variable contributing area (Spence et al., 2010; Fovet et al., 2015). However, most common hydrological models do not explicitly account for hysteresis, and often struggle to correctly simulate the

~~dry-to-wet~~ dry–to–wet transitions (de Boer-Euser et al., 2017). In our study, the simple dynamical systems approach assumes a fixed ~~storage-discharge~~ storage–discharge relation, with discharge being a function ~~of only~~ only of storage. In earlier studies where the concept was applied in a lumped fashion, this did not limit the performance of the concept, especially in the studies where the $g(Q)$ parameters were retrieved from the discharge observations (Kirchner, 2009; Teuling et al., 2010). In our case study, this also did not seem to be a limiting factor. We do expect the model performance to be affected in basins ~~with~~ exhibiting a strong hysteresis, yet we expect that one might define the $g(Q)$ function in a way that best captures the average behaviour of the rising and falling limbs.

Surface runoff as result of either infiltration excess or saturation excess ~~are both~~ is not explicitly accounted for. Surface runoff induced by ~~over-saturated~~ oversaturated soils is implicitly accounted for in the discharge sensitivity function, which will reach high sensitivity values with high discharge (i.e. storage). During precipitation events with high intensity, surface runoff can also occur when the soil's infiltration capacity cannot ~~meet~~ accommodate the precipitation intensity. The model might therefore underestimate some discharge peaks resulting from infiltration excess overland flow. However, we expect that the influence of this process is minor in humid catchments. Not accounting for this process relates back to the purpose of this model, where we do not focus on correctly representing catchments' internal fluxes, but instead focus on simulating the total discharge in a computationally efficient manner. It does, however, disqualify this concept for catchments with extreme precipitation events and the corresponding overland flow processes.

Furthermore, this model relies on the user to provide accurate actual evapotranspiration values as input for the model. As can be seen in Eq. (3), evaporation is directly taken into account in determining the change in storage. Due to the simplicity of the concept, the model does not contain any evaporation reduction processes as result of e.g. soil moisture stress. The model disables evaporation when discharge drops below a threshold, behaving like an ~~on-off~~ on–off switch, which is not how it is observed in reality. This reduction is most importantly preventing the model ~~to calculate~~ from calculating negative discharge values, while trying to behave like a real process.

The vectorized implementation limits the ability to transfer water between neighbouring pixels. This subsurface lateral flow is mostly driven by gravity, and is therefore expected to be most important in regions with large elevation differences. However, at the current recommend ~~spatial-temporal~~ spatial–temporal scale of application ($\sim$ 1 km^2 and 1 h), we expect lateral subsurface flow to be relatively unimportant. At smaller spatial scales or longer time scales, flow between pixels can become more important. To stay in line with the original scale of the simple dynamical systems approach, we do not recommend to further reduce the spatial resolution or to further increase the temporal resolution. Furthermore, most common hydrological models also do not account for flow between pixels (Liang et al., 1996; Arnold et al., 1998; Terink et al., 2015).

As proposed by Kirchner (2009), this concept can be used to "do hydrology backwards": infer precipitation from the discharge time series. Unfortunately, the distributed implementation of this concept complicates the use of this model to infer spatially distributed rainfall maps. Since the resulting discharge is an average of the integrated catchment response, it is impossible to infer the location of the precipitation event. This will become especially difficult in large basins, where the time ~~lag~~ lags induced by routing will become larger. A study by Pan and Wood (2013) describes a method to infer precipitation from

~~the stream flow~~ streamflow observations, though it requires a relatively ~~high~~ large number of observations. This method can potentially be applied to the model presented in this paper.

Finally, the current technical implementation of the model implies two other limitations. First, the routing scheme only induces a time lag on the discharge peak, and does not account for diffusion of the wave as it travels through the river network. For smaller basins, the effect of diffusion is only minor, but when simulating larger basins such as the Rhine, diffusion of the discharge wave will play a more important role. We recommend to use a more advanced routing scheme when applying the model to larger basins. The model can output a NetCDF file with runoff generated in each pixel per time step, which can easily be connected to other routing software such as SOBEK. Secondly, the current version of the model does not (yet) support multi-threading, meaning that the model currently only runs on a single thread. Spreading the computational load over multiple threads is expected to further reduce the time required to run the model, especially in large basins. To make the model more efficient in ~~single-thread~~ single thread mode, one can split the entire basin into several smaller sections and simultaneously run the model for each section. After the simulation is complete, the user can combine the several resulting pixel-wise runoff maps, and apply a routing scheme ~~on~~ to the entire basin.

7 Conclusions

The distributed simple dynamical systems (dS2) model is a new ~~rainfall runoff model with the aim~~ rainfall–runoff model that aims to simulate discharge in a computationally efficient manner. This model is based on the simple dynamical systems approach ~~as~~ proposed by Kirchner (2009), but has been modified to be applied in a distributed fashion, written in Python. In this way, the concept can be applied to larger basins, while respecting the original spatial and temporal scale of the concept, and also make use of high–resolution data. We have extended the concept with a snow module and a simple routing module. The snow module can be turned on or off, depending on the basin of interest, and the routing module is required to transport water through the basin towards the outlet. The flexibility in the discharge sensitivity ($g(Q)$) function and the resulting strong non-linearity make this model different from more common "bucket-type" models. As a result, dS2 is able to quickly simulate hydrological response at relatively high spatial ($\sim$1 km^2) and temporal ($\sim$1 hour) resolution.

Synthetic examples ~~demonstrated~~ demonstrate that, although dS2 solves the water balance implicitly, the model is able to accurately close the water balance. The response surface plots for all parameter combinations show that there are some correlations between the parameters, especially for the three discharge sensitivity parameters. However, we also showed that the parameters clearly influence different performance metrics, which provides the opportunity to reduce the calculation time of optimization algorithms. Finally, we have applied the model in the Thur basin as a case study to validate the performance of the model. Our model is able to correctly simulate discharge, both at the local scale (e.g. the Rietholzbach catchment in Switzerland, 3 km^2) and at the mesoscale (entire Thur basin in Switzerland, 1700 km^2), without a decrease in model performance as catchment size increases.

The distributed simple dynamical systems (dS2) model has several unique strengths compared to other ~~rainfall-runoff~~ rainfall–runoff models: (1) it is computationally efficient even at high temporal and spatial resolutions, (2) it only has five

parameters to calibrate, (3) two parameters have a clearly defined influence on the discharge time series making them easy to identify, and (4) the Python model code is open source and easily adjustable. The computational efficiency of this model creates the opportunity to answer different research questions. Since the model is able to simulate regions in relatively short amounts of time, performing sensitivity and uncertainty analyses with a large number of samples becomes feasible. Since the model is distributed, the sensitivity and uncertainty analyses can be performed both on a spatial and temporal basis. Furthermore, this model can be a valuable tool for educational purposes, to explain and directly show the effects of modifying parameters in distinct groups (e.g. hydrological response, snow, routing, evaporation). Overall, this makes dS2 a valuable addition to the already large pool of conceptual ~~rainfall-runoff~~ rainfall–runoff models, both for researchers ~~as~~ and for practitioners interested in large sample studies.

Code availability. The model code is open-source and is archived at the 4TU repository: https://doi.org/10.4121/uuid:cc8e0008-ab1f-43ee-b50d-24de01d2 (Buitink et al., 2019). The latest version of the dS2 model can be found on GitHub: https://github.com/JoostBuitink/dS2 (last access: 9 January 2020). On the same GitHub repository, a brief manual can be found which briefly explains what input data is required and how to set up and run the model. This manual uses the Rhine basin in Europe with the ERA5 dataset as an example.

Table A1. List of dS2 parameters. All parameters after rdf only influence either the numerical stability and/or the IO operations of the model, not the model output.

Name	Description	Example value	Unit
α	gQ parameter - intersect	-2.5	-
β	gQ parameter - slope	0.85	-
γ	gQ parameter - curvature	-0.010	-
ϵ	Reduction factor for evaporation	0.89	-
τ	Travel speed of water through the catchment	2	$\mathrm{m\,s^{-1}}$
T0	Critical temperature	0	$^{\circ}\mathrm{C}$
ddf	Degree day factor for snowmelt	2	$\mathrm{mm\,day^{-1}\,^{\circ}C^{-1}}$
rdf	Conversion factor for energy to snowmelt	0.26	$\mathrm{mm\,day^{-1}\,(W/r}$
Qt	Threshold below which evaporation is set to zero	10^{-4}	$\mathrm{mm\,h^{-1}}$
dt	Size of time step (Δt)	1	h
max_RAM	Maximum RAM the model is allowed to use	1024	MB
size_one_value	Size of a single value	4	bytes
LB	Factor controlling lower boundary for solver, where ~~$Q_t ! \leq Q_{t-1} \cdot \mathrm{LB}$~~ $Q_t \leq Q_{t-1} \cdot \mathrm{LB}$	10^{-4}	-
max_gQ_difference	Maximum allowed difference in gQ values	2	-
dt_reduction	Factor controlling the Δt reduction	0.15	-
min_extra_dt	Number of minimum extra Δt per Δt	5	-
max_extra_dt	Number of maximum extra Δt per Δt	50	-

Author contributions. JB, LAM and AJT conceived the idea behind dS2. JB developed the dS2 model, with valuable inputs from all co-authors. JB performed the synthetic experiments and case study, LAM and AJT helped with interpreting the results. JB prepared the manuscript, with significant contributions from all co-authors.

Acknowledgements. We would like to thank MeteoSwiss and Massimiliano Zappa for providing the forcing data for the case study.

References

Abramowitz, G.: Towards a Benchmark for Land Surface Models, Geophysical Research Letters, 32, https://doi.org/10.1029/2005GL024419, 2005.

Adamovic, M., Braud, I., Branger, F., and Kirchner, J. W.: Assessing the Simple Dynamical Systems Approach in a Mediterranean Context: Application to the Ardèche Catchment (France), Hydrology and Earth System Sciences, 19, 2427–2449, https://doi.org/10.5194/hess-19-2427-2015, 2015.

Adamovic, M., Branger, F., Braud, I., and Kralisch, S.: Development of a Data-Driven Semi-Distributed Hydrological Model for Regional Scale Catchments Prone to Mediterranean Flash Floods, Journal of Hydrology, 541, 173–189, https://doi.org/10.1016/j.jhydrol.2016.03.032, 2016.

Albergel, C., Dutra, E., Munier, S., Calvet, J.-C., Munoz-Sabater, J., de Rosnay, P., and Balsamo, G.: ERA-5 and ERA-Interim Driven ISBA Land Surface Model Simulations: Which One Performs Better?, Hydrology and Earth System Sciences, 22, 3515–3532, https://doi.org/10.5194/hess-22-3515-2018, 2018.

Arnold, J. G., Srinivasan, R., Muttiah, R. S., and Williams, J. R.: Large Area Hydrologic Modeling and Assessment - Part 1: Model Development, Journal of the American Water Resources Association, 34, 73–89, https://doi.org/10.1111/j.1752-1688.1998.tb05961.x, 1998.

Best, M. J., Abramowitz, G., Johnson, H. R., Pitman, A. J., Balsamo, G., Boone, A., Cuntz, M., Decharme, B., Dirmeyer, P. A., Dong, J., Ek, M., Guo, Z., Haverd, V., van den Hurk, B. J. J., Nearing, G. S., Pak, B., Peters-Lidard, C., Santanello, J. A., Stevens, L., and Vuichard, N.: The Plumbing of Land Surface Models: Benchmarking Model Performance, J. Hydrometeor., 16, 1425–1442, https://doi.org/10.1175/JHM-D-14-0158.1, 2015.

Beven, K.: Changing Ideas in Hydrology — The Case of Physically-Based Models, Journal of Hydrology, 105, 157–172, https://doi.org/10.1016/0022-1694(89)90101-7, 1989.

Beven, K.: How Far Can We Go in Distributed Hydrological Modelling?, Hydrol. Earth Syst. Sci., 5, 1–12, https://doi.org/10.5194/hess-5-1-2001, 2001.

Beven, K. J. and Kirkby, M. J.: A Physically Based, Variable Contributing Area Model of Basin Hydrology / Un Modèle à Base Physique de Zone d'appel Variable de l'hydrologie Du Bassin Versant, Hydrological Sciences Bulletin, 24, 43–69, https://doi.org/10.1080/02626667909491834, 1979.

Brauer, C. C., Teuling, A. J., Torfs, P. J. J. F., and Uijlenhoet, R.: Investigating Storage-Discharge Relations in a Lowland Catchment Using Hydrograph Fitting, Recession Analysis, and Soil Moisture Data, Water Resour. Res., 49, 4257–4264, https://doi.org/10.1002/wrcr.20320, 2013.

Brauer, C. C., Teuling, A. J., Torfs, P. J. J. F., and Uijlenhoet, R.: The Wageningen Lowland Runoff Simulator (WALRUS): A Lumped Rainfall–Runoff Model for Catchments with Shallow Groundwater, Geoscientific Model Development, 7, 2313–2332, https://doi.org/10.5194/gmd-7-2313-2014, 2014.

Brutsaert, W. and Lopez, J. P.: Basin-Scale Geohydrologic Drought Flow Features of Riparian Aquifers in the Southern Great Plains, Water Resources Research, 34, 233–240, https://doi.org/10.1029/97WR03068, 1998.

Buitink, J., Melsen, L. A., Kirchner, J. W., and Teuling, A. J.: The Distributed Simple Dynamical Systems (dS2) Model, https://doi.org/10.4121/uuid:cc8e0008-ab1f-43ee-b50d-24de01d2d0be, 2019.

Cash, J. R. and Karp, A. H.: A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides, ACM Trans. Math. Softw., 16, 201–222, https://doi.org/10.1145/79505.79507, 1990.

Clark, M. P. and Kavetski, D.: Ancient Numerical Daemons of Conceptual Hydrological Modeling: 1. Fidelity and Efficiency of Time Stepping Schemes, Water Resources Research, 46, https://doi.org/10.1029/2009WR008894, 2010.

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, https://doi.org/10.1029/2007WR006735, 2008.

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 Resour. Res., 51, 2498–2514, https://doi.org/10.1002/2015WR017198, 2015.

Comola, F., Schaefli, B., Ronco, P. D., Botter, G., Bavay, M., Rinaldo, A., and Lehning, M.: Scale-Dependent Effects of Solar Radiation Patterns on the Snow-Dominated Hydrologic Response, Geophysical Research Letters, 42, 3895–3902, https://doi.org/10.1002/2015GL064075, 2015.

Cornes, R. C., van der Schrier, G., van den Besselaar, E. J. M., and Jones, P. D.: An Ensemble Version of the E-OBS Temperature and Precipitation Data Sets, Journal of Geophysical Research: Atmospheres, 123, 9391–9409, https://doi.org/10.1029/2017JD028200, 2018.

de Boer-Euser, T., Bouaziz, L., De Niel, J., Brauer, C., Dewals, B., Drogue, G., Fenicia, F., Grelier, B., Nossent, J., Pereira, F., Savenije, H., Thirel, G., and Willems, P.: Looking beyond General Metrics for Model Comparison – Lessons from an International Model Intercomparison Study, Hydrol. Earth Syst. Sci., 21, 423–440, https://doi.org/10.5194/hess-21-423-2017, 2017.

Fovet, O., Ruiz, L., Hrachowitz, M., Faucheux, M., and Gascuel-Odoux, C.: Hydrological Hysteresis and Its Value for Assessing Process Consistency in Catchment Conceptual Models, Hydrology and Earth System Sciences, 19, 105–123, https://doi.org/10.5194/hess-19-105-2015, 2015.

Franchini, M. and O'Connell, P. E.: An Analysis of the Dynamic Component of the Geomorphologic Instantaneous Unit Hydrograph, Journal of Hydrology, 175, 407–428, https://doi.org/10.1016/S0022-1694(96)80018-7, 1996.

Fundel, F. and Zappa, M.: Hydrological Ensemble Forecasting in Mesoscale Catchments: Sensitivity to Initial Conditions and Value of Reforecasts, Water Resources Research, 47, https://doi.org/10.1029/2010WR009996, 2011.

Huuskonen, A., Saltikoff, E., and Holleman, I.: The Operational Weather Radar Network in Europe, Bull. Amer. Meteor. Soc., 95, 897–907, https://doi.org/10.1175/BAMS-D-12-00216.1, 2013.

Kavetski, D. and Clark, M. P.: Ancient Numerical Daemons of Conceptual Hydrological Modeling: 2. Impact of Time Stepping Schemes on Model Analysis and Prediction, Water Resources Research, 46, https://doi.org/10.1029/2009WR008896, 2010.

Kavetski, D. and Clark, M. P.: Numerical Troubles in Conceptual Hydrology: Approximations, Absurdities and Impact on Hypothesis Testing, Hydrological Processes, 25, 661–670, https://doi.org/10.1002/hyp.7899, 2011.

Kirchner, J. W.: Catchments as Simple Dynamical Systems: Catchment Characterization, Rainfall-Runoff Modeling, and Doing Hydrology Backward, Water Resources Research, 45, https://doi.org/10.1029/2008WR006912, 2009.

Kirkby, M. J.: Tests of the Random Network Model, and Its Application to Basin Hydrology, Earth Surface Processes, 1, 197–212, https://doi.org/10.1002/esp.3290010302, 1976.

Koster, R. D. and Suarez, M. J.: Soil Moisture Memory in Climate Models, J. Hydrometeor., 2, 558–570, https://doi.org/10.1175/1525-7541(2001)002<0558:SMMICM>2.0.CO;2, 2001.

Krier, R., Matgen, P., Goergen, K., Pfister, L., Hoffmann, L., Kirchner, J. W., Uhlenbrook, S., and Savenije, H. H. G.: Inferring Catchment Precipitation by Doing Hydrology Backward: A Test in 24 Small and Mesoscale Catchments in Luxembourg, Water Resour. Res., 48, W10 525, https://doi.org/10.1029/2011WR010657, 2012.

Kustas, W. P., Rango, A., and Uijlenhoet, R.: A Simple Energy Budget Algorithm for the Snowmelt Runoff Model, Water Resources Research, 30, 1515–1527, https://doi.org/10.1029/94WR00152, 1994.

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, 99, 14 415–14 428, https://doi.org/10.1029/94JD00483, 1994.

Liang, X., Wood, E. F., and Lettenmaier, D. P.: Surface Soil Moisture Parameterization of the VIC-2L Model: Evaluation and Modification, Global and Planetary Change, 13, 195–206, https://doi.org/10.1016/0921-8181(95)00046-1, 1996.

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.

Liu, Y., Hejazi, M., Li, H., Zhang, X., and Leng, G.: A Hydrological Emulator for Global Applications – HE v1.0.0, Geoscientific Model Development, 11, 1077–1092, https://doi.org/10.5194/gmd-11-1077-2018, 2018.

Lobligeois, F., Andréassian, V., Perrin, C., Tabary, P., and Loumagne, C.: When Does Higher Spatial Resolution Rainfall Information Improve Streamflow Simulation? An Evaluation Using 3620 Flood Events, Hydrology and Earth System Sciences, 18, 575–594, https://doi.org/10.5194/hess-18-575-2014, 2014.

Lyon, S. W. and Troch, P. A.: Hillslope Subsurface Flow Similarity: Real-World Tests of the Hillslope Péclet Number, Water Resources Research, 43, https://doi.org/10.1029/2006WR005323, 2007.

McDonnell, J. J., Sivapalan, M., Vaché, K., Dunn, S., Grant, G., Haggerty, R., Hinz, C., Hooper, R., Kirchner, J., Roderick, M. L., Selker, J., and Weiler, M.: Moving beyond Heterogeneity and Process Complexity: A New Vision for Watershed Hydrology, Water Resour. Res., 43, W07 301, https://doi.org/10.1029/2006WR005467, 2007.

Melsen, L. A., Teuling, A. J., van Berkum, S. W., Torfs, P. J. J. F., and Uijlenhoet, R.: Catchments as Simple Dynamical Systems: A Case Study on Methods and Data Requirements for Parameter Identification, Water Resour. Res., 50, 5577–5596, https://doi.org/10.1002/2013WR014720, 2014.

Melsen, L. A., Teuling, A., Torfs, P., Zappa, M., Mizukami, N., Clark, M., and Uijlenhoet, R.: Representation of Spatial and Temporal Variability in Large-Domain Hydrological Models: Case Study for a Mesoscale Pre-Alpine Basin, Hydrology and Earth System Sciences, 20, 2207–2226, https://doi.org/10.5194/hess-20-2207-2016, 2016a.

Melsen, L. A., Teuling, A. J., Torfs, P. J. J. F., Uijlenhoet, R., Mizukami, N., and Clark, M. P.: HESS Opinions: The Need for Process-Based Evaluation of Large-Domain Hyper-Resolution Models, Hydrology and Earth System Sciences, 20, 1069–1079, https://doi.org/10.5194/hess-20-1069-2016, 2016b.

Pan, M. and Wood, E. F.: Inverse Streamflow Routing, Hydrol. Earth Syst. Sci., 17, 4577–4588, https://doi.org/10.5194/hess-17-4577-2013, 2013.

Rodríguez-Iturbe, I. and Valdes, J. B.: The geomorphologic structure of hydrologic response, WAT. RESOUR. RES., 15, 1409–1420, https://doi.org/10.1029/WR015i006p01409, 1979.

Ruiz-Villanueva, V., Borga, M., Zoccatelli, D., Marchi, L., Gaume, E., and Ehret, U.: Extreme Flood Response to Short-Duration Convective Rainfall in South-West Germany, Hydrology and Earth System Sciences, 16, 1543–1559, https://doi.org/10.5194/hess-16-1543-2012, 2012.

Rupp, D. E. and Selker, J. S.: On the Use of the Boussinesq Equation for Interpreting Recession Hydrographs from Sloping Aquifers, Water Resources Research, 42, https://doi.org/10.1029/2006WR005080, 2006.

Rupp, D. E. and Woods, R. A.: Increased Flexibility in Base Flow Modelling Using a Power Law Transmissivity Profile, Hydrological Processes, 22, 2667–2671, https://doi.org/10.1002/hyp.6863, 2008.

Saltelli, A.: Making Best Use of Model Evaluations to Compute Sensitivity Indices, Computer Physics Communications, 145, 280–297, https://doi.org/10.1016/S0010-4655(02)00280-1, 2002.

Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., and Tarantola, S.: Variance Based Sensitivity Analysis of Model Output. Design and Estimator for the Total Sensitivity Index, Computer Physics Communications, 181, 259–270, https://doi.org/10.1016/j.cpc.2009.09.018, 2010.

Schoups, G., Vrugt, J. A., Fenicia, F., and van de Giesen, N. C.: Corruption of Accuracy and Efficiency of Markov Chain Monte Carlo Simulation by Inaccurate Numerical Implementation of Conceptual Hydrologic Models, Water Resources Research, 46, https://doi.org/10.1029/2009WR008648, 2010.

Sivapalan, M.: Pattern, Process and Function: Elements of a Unified Theory of Hydrology at the Catchment Scale, Pattern, Process and Function: Elements of a Unified Theory of Hydrology at the Catchment Scale, in: Encyclopedia of Hydrological Sciences, Encyclopedia of Hydrological Sciences, John Wiley & Sons, Ltd, John Wiley & Sons, Ltd, https://doi.org/10.1002/0470848944.hsa012, 2006.

Sobol′, I. M.: Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates, Mathematics and Computers in Simulation, 55, 271–280, https://doi.org/10.1016/S0378-4754(00)00270-6, 2001.

Spence, C., Guan, X. J., Phillips, R., Hedstrom, N., Granger, R., and Reid, B.: Storage Dynamics and Streamflow in a Catchment with a Variable Contributing Area, Hydrological Processes, 24, 2209–2221, https://doi.org/10.1002/hyp.7492, 2010.

Sutanudjaja, E. H., van Beek, R., Wanders, N., Wada, Y., Bosmans, J. H. C., Drost, N., van der Ent, R. J., de Graaf, I. E. M., Hoch, J. M., de Jong, K., Karssenberg, D., López López, P., Peßenteiner, S., Schmitz, O., Straatsma, M. W., Vannametee, E., Wisser, D., and Bierkens, M. F. P.: PCR-GLOBWB 2: A 5 Arcmin Global Hydrological and Water Resources Model, Geoscientific Model Development, 11, 2429–2453, https://doi.org/10.5194/gmd-11-2429-2018, 2018.

Tague, C. and Grant, G. E.: A Geological Framework for Interpreting the Low-Flow Regimes of Cascade Streams, Willamette River Basin, Oregon, Water Resources Research, 40, https://doi.org/10.1029/2003WR002629, 2004.

Terink, W., Lutz, A. F., Simons, G. W. H., Immerzeel, W. W., and Droogers, P.: SPHY v2.0: Spatial Processes in HYdrology, Geosci. Model Dev., 8, 2009–2034, https://doi.org/10.5194/gmd-8-2009-2015, 2015.

Teuling, A. J., Lehner, I., Kirchner, J. W., and Seneviratne, S. I.: Catchments as Simple Dynamical Systems: Experience from a Swiss Prealpine Catchment, Water Resources Research, 46, https://doi.org/10.1029/2009WR008777, 2010.

Troch, P. A., Troch, F. P. D., and Brutsaert, W.: Effective Water Table Depth to Describe Initial Conditions Prior to Storm Rainfall in Humid Regions, Water Resources Research, 29, 427–434, https://doi.org/10.1029/92WR02087, 1993.

Troch, P. A., Paniconi, C., and van Loon, E. E.: Hillslope-Storage Boussinesq Model for Subsurface Flow and Variable Source Areas along Complex Hillslopes: 1. Formulation and Characteristic Response, Water Resources Research, 39, https://doi.org/10.1029/2002WR001728, 2003.

van Osnabrugge, B., Weerts, A. H., and Uijlenhoet, R.: genRE: A Method to Extend Gridded Precipitation Climatology Data Sets in Near Real-Time for Hydrological Forecasting Purposes, Water Resources Research, 53, 9284–9303, https://doi.org/10.1002/2017WR021201, 2017.

van Osnabrugge, B., Uijlenhoet, R., and Weerts, A.: Contribution of Potential Evaporation Forecasts to 10-Day Streamflow Forecast Skill for the Rhine River, Hydrology and Earth System Sciences, 23, 1453–1467, https://doi.org/10.5194/hess-23-1453-2019, 2019.

Viviroli, D., Zappa, M., Gurtz, J., and Weingartner, R.: An Introduction to the Hydrological Modelling System PREVAH and Its Pre- and Post-Processing-Tools, Environmental Modelling & Software, 24, 1209–1222, https://doi.org/10.1016/j.envsoft.2009.04.001, 2009.

Weiler, M. and Beven, K.: Do We Need a Community Hydrological Model?, Water Resources Research, 51, 7777–7784, https://doi.org/10.1002/2014WR016731, 2015.

Winterrath, T., Mario, H., Junghänel, T., Klameth, A., Lengfeld, K., Walawender, E., Weigl, E., and Becker, A.: RADKLIM Version 2017.002: Reprocessed Gauge-Adjusted Radar Data, One-Hour Precipitation Sums, https://doi.org/10.5676/DWD/RADKLIM_RW_V2017.002, 2018.