Technical note: “Bit by bit”: A practical and general approach for evaluating model computational complexity vs. model performance

Elnaz Azmi[1], Uwe Ehret[2], Steven V. Weijs[3], Benjamin L. Ruddell[4], Rui A. P. Perdigão[5,6,7]

[1]Steinbuch Centre for Computing, Karlsruhe Institute of Technology - KIT, Karlsruhe, Germany
[2]Institute of Water Resources and River Basin Management, Karlsruhe Institute of Technology - KIT, Karlsruhe, Germany
[3]Department of Civil Engineering, University of British Columbia, Canada
[4]School of Informatics, Computing, and Cyber Systems, Northern Arizona University, U.S.A
[5]Meteoceanics Interdisciplinary Centre for Complex System Science, Vienna, Austria
[6]CCIAM, Centre for Ecology, Evolution and Environmental Changes, Universidade de Lisboa, Lisbon, Portugal
[7]Physics of Information and Quantum Technologies Group, Instituto de Telecomunicações, Lisbon, Portugal

Correspondence to: Elnaz Azmi (elnaz.azmi@kit.edu)

Abstract. One of the main objectives of the scientific enterprise is the development of well-performing yet parsimonious models for all natural phenomena and systems. In the 21st century, scientists usually represent their models, hypotheses, and experimental observations using digital computers. Measuring performance and parsimony of computer models is therefore a key theoretical and practical challenge for 21st century science. 'Performance' here refers to a model's ability to reduce predictive uncertainty about an object of interest. 'Parsimony' (or complexity) comprises two aspects: descriptive complexity - the size of the model itself which can be measured by the disk space it occupies -, and computational complexity - the model's effort to provide output. Descriptive complexity is related to inference quality and generality, computational complexity is often a practical and economic concern for limited computing resources.

In this context, this paper has two distinct but related goals: The first is to propose a practical method of measuring computational complexity by utility software 'Strace', which counts the total number of memory visits while running a model on a computer. The second goal is to propose the 'bit by bit' method, which combines measuring computational complexity by 'Strace', and measuring model performance by information loss relative to observations, both in bit. For demonstration, we apply the 'bit by bit' method to watershed models representing a wide diversity of modelling strategies (artificial neural network, auto-regressive, process-based, and other). We demonstrate that computational complexity as measured by 'Strace' is sensitive to all aspects of a model, such as the size of the model itself, the input data it reads, its numerical scheme and time-stepping. We further demonstrate that for each model, the bit counts for computational complexity exceed those for performance by several orders of magnitude, and that the differences among the models for both computational complexity and performance can be explained by their setup, and are in accordance with expectations.

We conclude that measuring computational complexity by 'Strace' is practical, and it is also general in the sense that it can be applied to any model that can be run on a digital computer. We further conclude that the 'bit by bit' approach is general in the sense that it measures two key aspects of a model in the single unit of bit. We suggest that it can be enhanced by additionally measuring a model's descriptive complexity - also in bit.

Keywords

Computational complexity, descriptive complexity, model performance, model evaluation, information, entropy

1 Introduction

1.1 The goals of Science

One of the main objectives of the scientific enterprise is the development of parsimonious yet well-performing models for all natural phenomena and systems. Such models should produce output in agreement with observations of the related real-world system, i.e. perform well in terms of accuracy and precision and overall 'rightness' (Kirchner, 2006). Another key aspect of evaluating such models is their complexity, i.e. they should be brief, elegant, explainable, understandable, communicable, teachable, and small. Mathematical analytical models - e.g. Newton’s Laws - represent an ideal type of model because they combine performance (high accuracy and precision when compared with experimental observations) with minimal yet adequate complexity (high elegance, brevity, and communicability). Another key aspect of model complexity is how efficiently a model produces its output. This is especially relevant for large models used in operational settings, where computation times are an issue. In Fig. 1 panel (a), these key aspects of model evaluation are referred to as 'descriptive complexity', 'computational complexity' and 'performance'. A simple example to illustrate their relation: Suppose we want to bake a cake; then the length of the recipe measures its descriptive complexity, the time or effort it takes to actually prepare the cake by following the recipe instructions measures its computational complexity, and the (dis-) agreement of our cake with the gold standard cake from the pastry shop measures its performance.

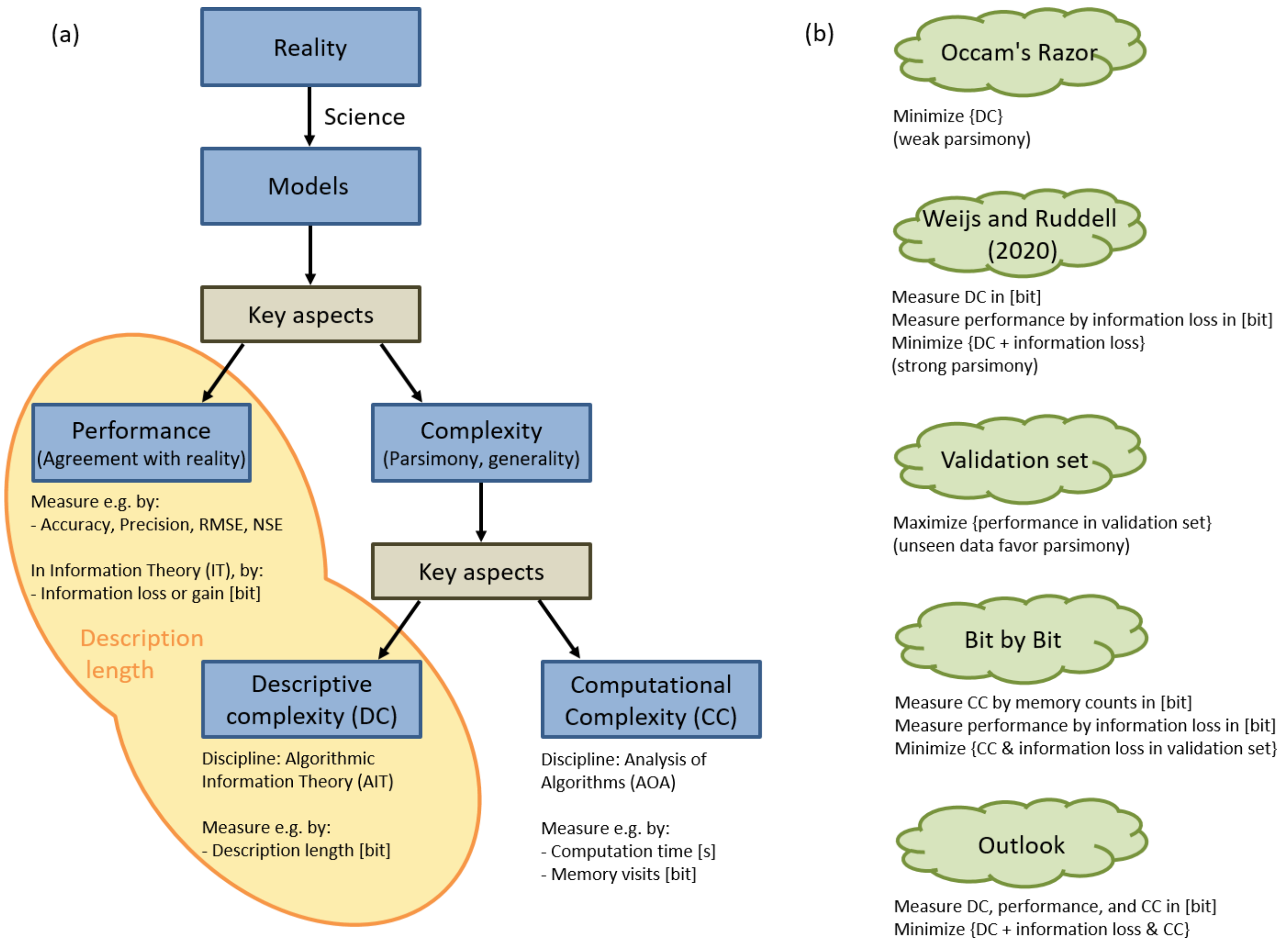
(a)
Reality
Science
Models
Key aspects
Performance
(Agreement with reality)
Measure e.g. by:
- Accuracy, Precision, RMSE, NSE
In Information Theory (IT), by:
- Information loss or gain [bit]
Description length
Complexity
(Parsimony, generality)
Key aspects
Descriptive complexity (DC)
Discipline: Algorithmic Information Theory (AIT)
Measure e.g. by:
- Description length [bit]
Computational Complexity (CC)
Discipline: Analysis of Algorithms (AOA)
Measure e.g. by:
- Computation time [s]
- Memory visits [bit]
(b)
Occam's Razor
Minimize {DC}
(weak parsimony)
Weijs and Ruddell (2020)
Measure DC in [bit]
Measure performance by information loss in [bit]
Minimize {DC + information loss}
(strong parsimony)
Validation set
Maximize {performance in validation set}
(unseen data favor parsimony)
Bit by Bit
Measure CC by memory counts in [bit]
Measure performance by information loss in [bit]
Minimize {CC & information loss in validation set}
Outlook
Measure DC, performance, and CC in [bit]
Minimize {DC + information loss & CC}

Figure 1. (a): Aspects of model evaluation. (b): Guiding principles for model development.

1.2 Guidelines for developing parsimonius models

Several approaches exist to guide model development (Fig. 1 panel (b)), and they differ by the emphasis they put on each of the three previously discussed key aspects. We will in the following briefly describe some of these guidelines to provide the background for the 'bit by bit' approach suggested in the paper.

Occam's Razor, a bedrock principle of science, argues that the least descriptively complex model is preferable, at a given level of predictive performance that is adequate to the question or application at hand. Occam's Razor is a guideline to promote models that describe well patterns in the data and to distil laws that allow effective compression of experimental data; also it is a guideline for inference. In the framework of Algorithmic Information Theory (AIT) (Kolmogorov, 1968; Solomonoff 1964, 1978; Chaitin 1966), descriptive complexity of a model is measured by its size expressed in bit, when stored as instructions for a computer. It is noteworthy that Occam's razor puts an emphasis on descriptive complexity, considers performance as a side condition, but is completely independent of any practical considerations such as limited storage space or computing power, i.e. it ignores computational complexity. So while Occam promotes models that achieve effective compression of experimental data, compression for the sake of meeting constraints in a storage limited world is not the primary goal, but rather the reverse: finding the shortest description is the process of inference, achieved by distilling patterns from data in order to find general predictive laws.

Weijs and Ruddell (2020), call Occam's parsimony a 'weak parsimony' because it identifies a set of parsimonious models rather than a single most-parsimonious model. They further argue that a single, 'strongly parsimonious' model could be identified by considering, in addition to model descriptive complexity, also model performance, and to express them in the language of AIT as two additive terms which together are the description length of the data in bits (yellow shaded area in Fig. 1). A strongly parsimonious model in the terms of Weijs and Ruddell (2020) perfectly (or losslessly) reproduces experimental observations in the smallest number of bits, after adding together the compressed size of the model and the compressed corrections needed to adjust the model's predictions to equal the observations. Such a model balances minimum model size and minimum information loss, and maximum generalizability outside the observed datasets used to construct and test the model. The latter claim is based on insights from AIT, where shorter descriptions have been shown to be more likely to be generalizable. Detailed explanations on this topic are given in Weijs and Ruddell (2020) and references therein. The approach proposed by Weijs and Ruddell (2020), drawing on the minimum description length principle (Rissanen, 1978; Grünwald, 2007) not only has the advantage of favouring models with a good trade-off between descriptive complexity and performance: Applying a single measure, expressed in bits, to quantify both of these aspects also offers the advantage of rigor and generality over more contextually defined performance measures, such as Root Mean Square Error, Nash-Sutcliffe Efficiency (Nash and Sutcliffe, 1970), Kling-Gupta Efficiency (Gupta et al., 2009), Akaike Information Criterion (Akaike, 1974), or Bayesian Information Criterion (Schwarz, 1978), to name just a few (more in Bennett et al., 2013). This more generalized strategy helps guiding model preference, especially in automated environments for learning models from data.

Validation set approaches are a standard procedure in hydrological model development. Among a set of competing models, the model is preferred that performs best on data unseen during model parameter estimation. The fact that model performance is evaluated on a validation set promotes models that are general, i.e. models that have captured the essential workings of the natural system they represent, and demotes models overfitted to the calibration data, which are likely to be models with unnecessarily high descriptive complexity.

In summary, both Occam's razor and the extension proposed by Weijs and Ruddell (2020) are designed with a focus on inference, i.e. on distilling small and universal laws from experimental data, while the focus of validation set approaches is mainly on performance. In neither of them the model's effort of actually making its predictions is directly considered. This effort, however, can be an important quality of a model in settings where computing resources are limited. In earth science modelling, this is the rule rather than the exception for the following reasons: i) scales of earth systems cannot be separated easily and in some cases not at all, so even for local questions it may be necessary to simulate large systems at a level of great spatio-temporal detail; ii) calibration of model parameters from data needs many repeated model runs for parameter identification; iii) models used in optimal decision making require repeated use to identify the optimal alternative. The efficiency at which models generate their output is subject of the discipline of Analysis of Algorithms (AOA). In AOA, it is referred to as computational complexity, and can be measured in terms of two finite resources that are needed for computation: time and/or space. Time-complexity relates to the time a computer needs to arrive at the result. Time complexity can be measured in terms of clock cycles, and often it is the scaling with the input size that is of interest. Space-complexity relates to the memory used, i.e. the total number of binary transistor states read during the execution of the program. As for descriptive complexity, these reads can be interpreted as answers to Yes/No questions, and can be measured in bit.

1.3 Scope and goals of this paper

In the context of the guidelines for model development discussed in the previous section, this paper has two distinct but related goals: The *first goal* is to propose a practical method of measuring computational complexity by 'Strace', a troubleshooting and monitoring utility for computer programs. 'Strace' counts the total number of memory visits while running a model on a computer. The counting is sensitive to all aspects of the model, such as the size of the model itself, the size of the input data it reads, the model's numerical scheme, time-stepping, runtime environment, etc. The *second goal* is to demonstrate how measuring computational complexity by 'Strace' can be combined with either a validation set approach or the approach suggested by Weijs and Rudell (2020) to jointly evaluate all key aspects of a model - descriptive complexity, computational complexity, and performance. We use a validation set approach here, as hydrologists are familiar with it, but adopt from Weijs and Ruddell (2020) to express model performance by information loss in bit. The 'bit by bit' approach as presented here therefore consists of explicitly evaluating a model in terms of computational complexity and performance, both in bit. Descriptive complexity is implicitly considered by the validation set approach. Measuring computational complexity by 'Strace' is general in the sense that it can be applied to any model that can be run on a digital computer; the 'bit by bit' approach is general in the sense that it measures two key aspects of a model in the single unit of bit.

For demonstration, we run hydrological models of various types (artificial neural network, auto-regressive, simple and more advanced process-based, and both approximate and exact restatements of experimental observations) that all aim to perform the same task of predicting discharge at the outlet of a watershed. Akin to Weijs and Ruddell (2020), we examine possible trade-offs between computational complexity vs. information loss. It is important to note that the purpose of the model comparison here is not primarily to identify the best among the different modelling approaches, rather it serves as a demonstration how 'Strace' is sensitive to all facets of a model, and how differences among the models can be explained by their setup and are in accordance with expectations. In short, the aim is to provide a proof-of-concept.

The remainder of the manuscript is structured as follows: In section two, we describe the real-world system we seek to represent (a small alpine watershed in western Austria), the range of models we use for demonstration of the 'bit by bit' concept, and the implementation environment and criteria for measuring model performance and computational complexity.

In section three, we present and compare the models in terms of these criteria and illuminate differences between descriptive and computational complexity. In section four, we draw conclusions, discuss the limitations of the approach, and provide directions for future work.

1.4 Uses of 'complexity' in the hydrological sciences

A brief note on the uses of the term 'complexity', in this paper and in the hydrological sciences in general: In this paper, we use it in very specific ways to refer to different characteristics of a model. We have adopted the term 'descriptive complexity' from Algorithmic Information Theory to express the parsimony of a model by its size in bit when stored on a computer, and the term 'computational complexity' from Analysis of Algorithms to express the efficiency at which a model generates its output by the number of memory visits during program execution. In the hydrological sciences in general, 'complexity' is most often used in the wide sense of its dictionary definition to refer to 'systems consisting of many, different but related parts that are hard to separate, analyse, explain or understand'. Hydrological systems have been described and analysed in terms of their complexity by Jovanovic et al. (2017), Ossola et al. (2015), Bras (2015) and others; similarly, hydrological time series complexity was investigated by e.g. Engelhardt et al. (2009). Complexity measures have been used for classification of hydrological systems by Pande and Moayeri (2018), Sivakumar and Singh (2012) and Sivakumar et al. (2007). In this context, many different complexity measures have been proposed based on e.g. information entropy (Zhou et al., 2012; Castillo et al., 2015), wavelets (Sang et al., 2011), correlation dimension of system output (Sivakumar and Singh, 2012), and dynamic source analysis (Perdigão 2018; Perdigão et al. 2019). In hydrological modelling, 'model complexity' most often refers to the number of processes, variables, or parameters a model comprises, and many authors have investigated the relation of model complexity and predictive performance (Gan et al., 1997; Schoups et al., 2008; Arkestein and Pande, 2013; Forster et al., 2014; Finger et al., 2015; Orth et al., 2015). Our research contributes to the large existing body of complexity studies in hydrology, and we believe that by expressing all key aspects of computer-based models – performance, descriptive complexity and computational complexity – in the single general unit of bit offers an attractive avenue for comprehensive model evaluation and optimization.

2 Methods

2.1 The real-world system: a watershed in Austria

The real-world system we seek to represent with our models is the Dornbirnerach catchment in western Austria. Upstream of river gauge Hoher Steg (Q_Host), the target of our model predictions, the catchment covers 113 km². The catchment's rainfall-runoff dynamics reflect its alpine setting: Winter snow accumulation, spring snowmelt, high and intensive summer rainfall and, due to the steep terrain, rapid rainfall-runoff response. The meteorological dynamics of the system are represented by precipitation observations at a single rain gauge, Ebnit (P_Ebnit), located in the catchment centre. Both time series are available in hourly resolution for ten years (1996/01/01 – 2005/12/31). No other dynamical or structural data were used for model set up. While this would be overly data-scarce if we wanted to build the best possible hydrological model, we deemed it adequate for the aim of this study, i.e. demonstration of the bit-by-bit approach.

2.2 Models

We selected altogether eight modelling approaches with the aim of covering a wide range of model characteristics such as type (ignorant, perfect, conceptual-hydrological and data-driven), structure (single and double linear reservoir), numerical scheme (explicit and iterative) and precision (double and integer). The models are listed and described in Table 1, additional information is given in Fig. 2. We trained/calibrated each model on a five-year calibration period (1996/01/01 – 2000/12/31) and validated them in the five-year validation period (2001/01/01 – 2005/12/31).

Table 1. Models used in the study and their characteristics.

ID	Description	Time stepping *dt*	Variable precision	Numerical scheme	Training data	Data for running the model
Model-00	An (almost) ignorant model, which predicts for each time step the observed time series mean (4.86 m^3/s)	1 h	double	--	Q_Host	--
Model-01	A perfect model representing full prior knowledge contained in the experimental observations. For each time step, the observed value of Q_Host is read as input and provided as output.	1 h	double	--	Q_Host	Q_Host(*t*)
Model-02	A simple conceptual hydrological model, representing the catchments' rainfall-runoff behaviour by a single linear reservoir (Fig. 2, panel (a)) with a single parameter - K -, and a single state variable - S. K was found by calibration (K = 64 h). Time stepping is d*t* = 1 h, all variables are double precision, and the numerical scheme is explicit.	1 h	double	explicit	Q_Host P_Ebnit	P_Ebnit (*t*)
Model-03	Same as Model-02, but time stepping is d*t* = 1 min	1 min	double	explicit	Q_Host P_Ebnit	P_Ebnit(*t*)
Model-04	Same as Model-02, but all variables are integer precision only	1 h	integer	explicit	Q_Host P_Ebnit	int(P_Ebnit(*t*))
Model-05	A more advanced conceptual model (Fig. 2, panel (b)). Precipitation input is split by an intensity threshold - T - (3.5 mm/h), and enters two linear reservoirs - K1 - (10 h) and - K2 - (80 h). All parameters were found by calibration. Time stepping, variable precision, and numerical scheme are the same as in Model-02.	1 h	double	explicit	Q_Host P_Ebnit	P_Ebnit(*t*)
Model-06	Same as Model-02, but the numerical scheme is iterative.	1 h	double	iterative	Q_Host P_Ebnit	P_Ebnit(t)
Model-07	A simple third-order autoregressive model, which predicts Q_Host(*t*) by a linear combination of	1 h	double	--	Q_Host	Q_Host(*t*-3) Q_Host(*t*-2)

	previous observations in the form $Q(t) = c_0 + c_1 Q(t-1) + c_2 Q(t-2) + c_3 Q(t-3)$ All coefficients were found by calibration (c_0 = 0.0536, c_1 = 1.9916, c_2 = -1.3130, c_3 = 0.3104). Testing models of various order we found that adding observations beyond lag-3 improved predictive power only marginally				P_Ebnit	Q_Host(*t*-1)
Model-08	A long short-term memory (LSTM) that is an artificial recurrent neural network (LSTM) with a single hidden layer of 5 neurons and rolling window of size 20 along the time axis, using P_Ebnit(*t*) as input to predict Q_Host(*t*). The model is written in Python with the 'Keras' library. In the learning process, it uses the 'adam' optimizer with the loss function 'mean squared error'.	1 h	double	--	Q_Host P_Ebnit	P_Ebnit(t)

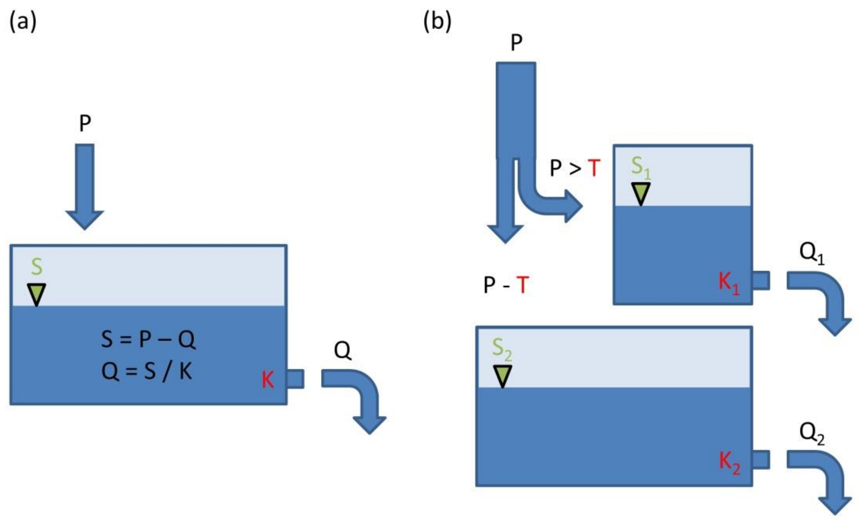

Figure 2. (a): Model-02, a single linear reservoir with state variable S and retention constant K. The reservoir is replenished by precipitation P and drained by discharge Q. (b): Model-05, with two linear reservoirs. Precipitation input is split by intensity threshold T.

2.3 Implementation environment

All models were implemented as Python scripts running on Python 3.6 with the installed packages Numpy, Pandas, Scipy, Keras and H5py. The experiments were done on a computer running Red Hat Enterprise Linux Server release 7.4 on a 16-core Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHz processor.

2.4 Measures of model performance and computational complexity

All models were evaluated in terms of the two criteria described in the introduction: performance, i.e. the model's ability to reduce predictive uncertainty about the target, and computational complexity, i.e. the effort required to make the model generate a prediction about the target. Similar to Weijs and Ruddell (2020), we express both quantities in bits, to assure generality and comparability.

2.4.1 Model performance

As in Weijs and Rudell (2020), we express model performance in terms of information losses. In information theory, information is defined as the negative logarithm of the probability p of an event. Information entropy $H(X)$ is defined as the expected or average value of information (Eq. 1) of a specific value of a data set $X = \{x_1, x_2, \dots, x_n\}$.

$$H(X) = \boldsymbol{E}[I(x)] = -\sum_{x \in X} p(x)\ log_2\, p(x) \tag{1}$$

Entropy is a measure of our uncertainty about the outcome of a random draw from a distribution *before* it is revealed to us, when all we know a priori is the data distribution. If we know the outcome beforehand (e.g. because we cheated), then the a priori known data distribution reduces to a Dirac function with $p = 1$ for the outcome and $p = 0$ everywhere else. The entropy of such a distribution - and with it our uncertainty - is zero. In model performance evaluation, we can use this 'perfect prediction' case as a benchmark to compare other states of prior knowledge against in terms of added uncertainty (or information lost). In the case described above, where all we know a priori is the data distribution, the information loss compared to the benchmark case equals the entropy of the distribution. In other cases, we may have useful side information – e.g. predictions of a model, which reduces information loss. In such a case, information loss can be quantified by conditional entropy (Eq. 2), where X represents the target and Y the model predictions (= predictor), y is a particular prediction, and H($X|Y$) is conditional entropy in bit.

$$H(X|Y) = \sum_{y \in Y} p(y)\, H(X|Y = y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y)\ log_2\, p(x|y) \tag{2}$$

Note that for models providing single-valued (deterministic) predictions, in order to construct a predictive distribution for which we can calculate an entropy, we have to assume that our state of knowledge for each prediction is given by the subset of observations jointly occurring with the particular prediction (the conditional distribution of X for a particular y). If models would give probabilistic predictions, we could directly employ a relative entropy measure such as Kullback-Leibler divergence (Kullback and Leibler, 1951; Cover and Thomas, 2006), which would lead to fairer assessments of information loss (Weijs et al., 2010). However, models that directly output probabilistic predictions are not yet a standard in hydrology.

Alternatively to measuring information losses of model predictions compared to an upper benchmark - the observations – as described above, it is also possible to measure information gains compared to a lower benchmark – the entropy of a uniform distribution - which expresses minimum prior knowledge. Weijs and Ruddell (2020), which we refer to throughout the text, used information losses because they directly translate to a description length. For reasons of comparability we applied the same concept here.

To avoid fitting of theoretical functions to the empirical data distributions, we calculated conditional entropy of discrete (binned) distributions, i.e. normalized empirical histograms. Choice of the binning scheme has important implications on the values of the information measures derived from the binned distributions: While the lower bound for entropy, $H = 0$ for a Dircac distribution, is independent of the number of bins n, the upper bound, $H = \log_2(n)$ for the maximum-entropy uniform distribution is a function of n. Choice of n is typically guided by the objective to balance resolution and sufficiently populated bins, and different strategies have been proposed e.g by Knuth (2013), Gong et al. (2014) and Pechlivanidis et al. (2016). We applied uniform binning as it introduces minimal prior information (Knuth, 2013) and as it is simple and computationally efficient (Ruddell and Kumar, 2009). We uniformly split the value range of 0 - 150 m^3/s, which covers all observed and simulated values of Q_Host (0.05 – 137 m^3/s) into 150 bins of 1 m^3/s width each. Compared to the typical error associated with discharge measurements in small, alpine rivers, which may be as high as 10%, we deemed this an adequate resolution which neither averages away the data-intrinsic variability nor fine-grains to resolutions potentially dominated by random errors.

When calculated in the described manner, a lower bound and two upper benchmarks for the values of conditional entropy can be stated: If the model perfectly predicts the true target value, it will be zero. Non-zero values of conditional entropy quantify exactly the information lost by using an imperfect prediction. If predictor and target are independent, the conditional entropy will be equal to the unconditional entropy of the target, which in our case is H(Q_Host) = 3.46 bit. If in the worst case there would be no paired data of target and predictors to learn from via model calibration, and the physically feasible range of the target data would be the only thing known a priori, the most honest guess about the target value would be a uniform (= maximum entropy) distribution. For the 151 bins we used, the entropy of a uniform distribution is $H_{uniform} = \log_2(151) = 7.23$ bit.

2.4.2 Model computational complexity

We quantify computational complexity by the total number of memory read visits (in bit) on a computer while running the model. In the context of Information Theory, these bit counts and the bits measuring model performance by conditional entropy in the previous section can both be interpreted in the same manner as a number of binary Yes/No questions that were either already asked and answered during the model run (in the former case) or still need to be asked (in the latter case) in order to fully reproduce the data.

Counting memory visits while running a computer program can be conveniently done by 'Strace', a troubleshooting and monitoring utility for Linux (see http://man7.org/linux/man-pages/man1/strace.1.html). It is a powerful tool to diagnose, debug and trace interactions between processes and the Linux kernel (Levin and Syromyatnikov, 2018). 'Strace' is executable along with running code in any programming language like python, C ++ or R. We instructed 'Strace' to monitor our test models written in Python by counting the total number of bytes read during the model execution from a file stored in the file system into a buffer, and the total number of bytes written from a buffer into a file stored in the file system. A buffer is a temporary data storing memory (usually located in the RAM) that prevents I/O bottleneck and speeds up the memory access. These counts reflect the entire effort of the model to produce the desired output: Reading input files, writing output files, reading the program itself and all system functions called during its execution, efforts of numerical iteration within the program as well as efforts to read and write state variables during runtime. Hence, 'Strace' will penalize models which require large amounts of forcing data, run on high-resolution time stepping or spatial resolution, or apply unnecessarily high-iterative numerical schemes. In short, 'Strace' evaluates all memory-related components of a model in the widest sense.

To evaluate the reproducibility of the countings, we repeated each model run 100 times, clearing the memory cache between individual runs. As the countings were in fact very close, we simply took the average of all runs as a single value representing model computational complexity. The main steps of applying 'Strace' in our work were as follows:

1. We traced the read() and write() system calls of the models while executing their code in Python and wrote them into a target log file running the following command in Linux commandline: 'strace -o target.log -e trace=read/write python model.py', where 'strace' is the executable tool, '-o target.log' is the option to set our log file path, '-e trace=read' traces the read() system call that returns the number of bytes read from the required files during the model execution into the system buffer, 'python' is the path to executable python program and 'model.py' is the path to our model code. Additionally, we used '-e trace=write' to trace the write() system call that returns the number of bytes written from the system buffer into the output file.
2. After generating the target log file, we calculated the sum of all read operations from the target log file running the following command: 'cat target.log | awk 'BEGIN {FS="="}{sum += $2} END {print sum >> "read_sum.txt"}', where 'cat' reads the target.log file, 'awk' scans the file and sums up the returned value of each read() and writes in the 'sum' variable, 'print sum' writes the sum value into a file called 'read_sum.txt'. Similarly, we summarized all write operations in a file. The sum of these read and write values is the total number of bytes which presents the evaluation of our model.

3 Results and discussion

As stated already previously, the purpose of the model comparison presented here is not primarily to identify the best among a set of competing models for a particular purpose. Rather, it is intended as a demonstration and proof-of-concept of how 'Strace' is sensitive to all facets of a model, how 'Strace' and the 'bit by bit' concept are applicable to a wide range of models, and how differences among the models in terms of computational complexity and performance can be explained by their setup.

3.1 Simulation vs. experimental observation

In Fig. 3, observed precipitation at Ebnit and observed and simulated discharge time series of all models at gauge Hoher Steg are shown for a rainfall-runoff event in June 2002, which lies within the validation period. The observed hydrograph (bold blue) shows a flood peak of 71 m³/s due to a 14-hour rainfall event. The ignorant Model-00 (black) is incapable of reproducing these dynamics and remains at its constant mean value prediction. Model-01 (light green) as expected perfectly matches the observations, and likewise the AR-3 Model-07 (pink) shows almost perfect agreement. The single-bucket Model-02 (purple) overall reproduces the observed rise and decline of discharge, but fails in the details: The rise is too slow and too small, and so is the decline. Apparently, a single linear reservoir cannot adequately represent the catchment's hydrological behaviour, irrespective of the time stepping and the numerical scheme: Discharge simulations by the high-resolution Model-03 (yellow) and the iterative Model-06 (dark green) are almost identical to that of Model-02. Data precision however does play a role: Model-04 (olive), identical to Model-02 except for a switch from double to integer precision of all variables, shows markedly worse performance. The hydrograph is only coarsely reproduced by a two-step series. From all linear reservoir models, the two-bucket Model-05 (brown) performs best, correctly reproducing the overall course of the event. The LSTM Model-08 (red) provides a good representation of the event rise, recession and peak discharge magnitude, but shows a delayed response with a lag of about three hours.

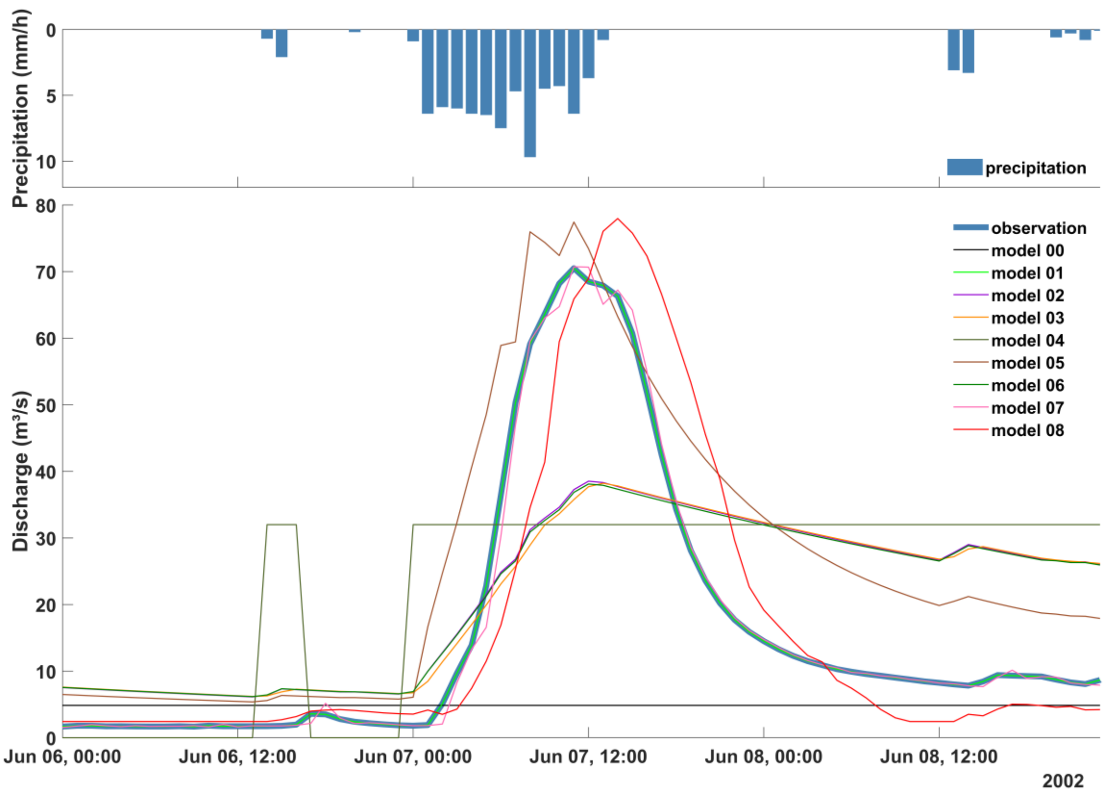

Figure 3. Top: Observed precipitation at Ebnit. Bottom: Observed discharge at gauge Hoher Steg and simulations thereof by Model-00 to Model-08 for a rainfall-runoff event in June 2002.

3.2 Performance vs. computational complexity

In the previous section we discussed model performance in terms of hydrologically informed visual comparison of observed and simulated hydrographs. Now we will evaluate the models in terms of both model performance and model computational complexity. Model performance is expressed as the remaining uncertainty, at each time step, about the observed data D given the related model simulation M by conditional entropy H(D|M) as described in section 2.4.1. Model computational complexity is expressed as the total number of memory read and write visits during model execution as counted by 'Strace'. For easier interpretation, we show average computational complexity per time step by dividing the total number of visits by the length of the validation period (43802 time steps).

Fig. 4 shows computational complexity and performance of all models. The theoretical optimum of zero information loss despite zero modelling effort lies in the lower left corner. Model-01, which simply reproduces the observations, came closest to this optimum, with perfect model performance (zero information loss), as to be expected. The mean Model-00, also as to be expected and just as in section 3.1, shows the worst performance of all models, but at least low computational complexity (1776 bits). The single-bucket Model-02 requires a higher computational effort (1797 bits), but model performance improves considerably. This is not the case for high-time-resolution Model-03, which, compared to Model-02, requires large computational efforts (4217 bits) without being rewarded by better model performance; a disadvantage not visible in Fig. 3. The poor performance of low-precision Model-04 however is visible in both Fig. 3 and Fig. 4, and the associated reduction of computational complexity (1755 bits) is only minor. Interestingly, the conceptually advanced two-bucket Model-05 performs similar to the single-bucket Model-02 both in terms of performance and computational complexity, while from the visual evaluation in Fig. 3, there was a clear advantage for Model-05. A possible explanation is that when measuring model performance by conditional entropy, it is not the closeness in value between model prediction and observation that indicates a good model, but rather the unambiguousness of the mapping between the two, which seems to be comparable for Model-02 and Model-05. Contrary to our expectations, the iterative Model-06 required hardly higher computational efforts than its non-iterative counterpart Model-02. The reason lies in the pronounced autocorrelation of the hydrological system response, such that in just a few cases - mainly at the onset of floods - iterations were actually needed to satisfy the chosen iteration precision limit of 0.001. The autoregressive Model-07 performs well (second-best, only outperformed by the perfect Model-01), and with respect to computational complexity (1817 bits) it is comparable to most other models. Obviously, a lot of information about discharge is contained in its own recordings of the immediate past, and this information can be tapped without much effort by an autoregressive model. The LSTM Model-08, shows comparable performance to the other models, but it is quite inefficient: For predicting a single time step, it visits 6083 bits of memory.

In the lower left corner of Fig. 4, a black square indicates a loose upper bound of the descriptive complexity of a single recording of our target discharge series Q_Host. The value (18.8 bit) was calculated by simply dividing the size of the Q_Host validation dataset by the number of time steps. This represents the raw size of a single data point in the series, without any compression, and if we want we can compare it to the computational effort of *generating* a single data point by any of the models. Clearly, the descriptive complexity is much smaller than the computational complexity.

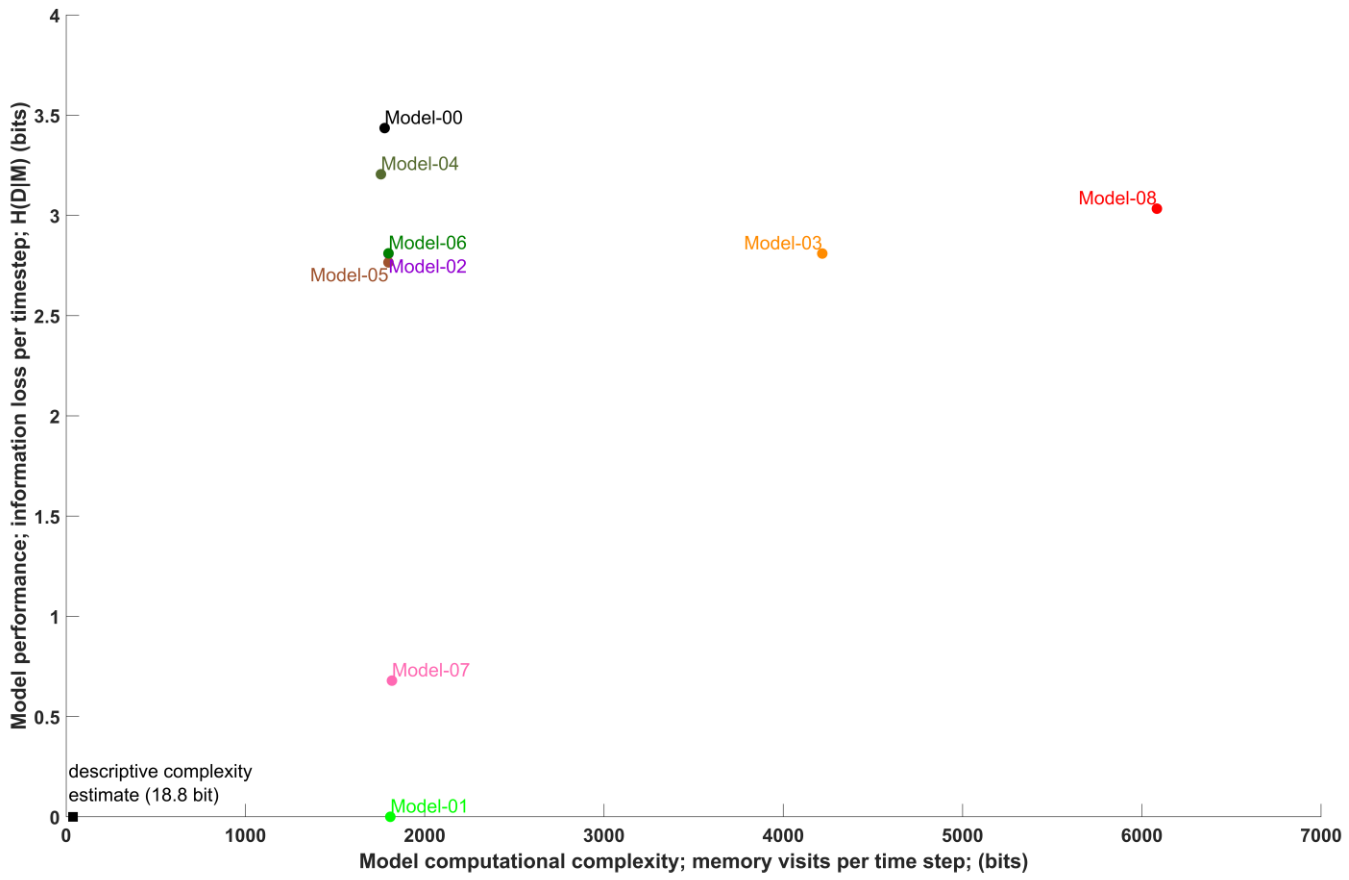

Figure 4. Model performance expressed by its inverse, information loss per time step, measured by conditional entropy in bits vs. model computational complexity measured by the average number of memory visits per time step in bits for Model-00 to Model-08.

4 Summary and conclusions

We started this paper by stating that one of the main objectives of the scientific enterprise is the development of well-performing yet parsimonious models for natural phenomena and systems, that models nowadays are mainly computer models, and that three key aspects for evaluating such models are descriptive complexity, computational complexity, and performance. We continued by describing several paradigms to guide model development: Occam's razor puts an emphasis on descriptive complexity, considers performance as a side condition, but it ignores computational complexity; Weijs and Ruddell (2020) express both model performance and descriptive complexity in bit, and by adding the two obtain a single measure for what they call 'strong parsimony'; validation set approaches focus on performance, and promote general and parsimonious models only indirectly by evaluating models on data not seen during calibration. Neither of these approaches directly incorporates computational complexity. We suggested to close this gap by 'Strace', a troubleshooting and monitoring utility, which measures computational complexity by the total number of memory visits while running a model on a computer. We further proposed the 'bit by bit' method, which combines measuring computational complexity by 'Strace', and measuring model performance by information loss relative to observations, all in bit, akin to Weijs and Ruddell (2020).

For a proof-of-concept, we applied the 'bit by bit' method in combination with a validation set approach - to also consider descriptive complexity, if only indirectly - at the example of a range of watershed models (artificial neural network, autoregressive, simple and advanced process-based with various numerical schemes). From the tested models, a third-order autoregressive model provided the best trade-off between computational complexity and performance, while the LSTM and a conceptual model operating in high temporal resolution showed very high computational complexity. For all models, computational complexity (in bit) exceeded the missing information (in bit) expressing model performance by about three orders of magnitude. We also compared a simple upper bound of descriptive complexity of the target data set to model computational complexity: The latter exceeded the former by about two orders of magnitude. Apart from these specific results, the main take-home messages from this proof-of-concept application are that i) measuring computational complexity by 'Strace' is general in the sense that it can be applied to any model that can be run on a digital computer; ii) 'Strace' is sensitive to all aspects of a model, such as the size of the model itself, the input data it reads, its numerical scheme and time-stepping; iii) the 'bit by bit' approach is general in the sense that it measures two key aspects of a model in the single unit of bit, such that they can be used together to guide model analysis and optimization in a pareto trade-off manner in the general setting of incremental learning. It can be useful especially in operational settings where the speed of information processing is a bottleneck. Unlike approaches to estimate computational complexity via model execution time, the bit counting by 'Strace' is unaffected by other ongoing processes on the computer competing for CPU time. This increases reproducibility and unambiguousness of the results. The 'bit by bit' approach can help promoting better model code in two ways: Computational complexity is sensitive to poor (inefficient) coding, performance is sensitive to wrong (erroneous) coding. This is relevant as computer models in the earth sciences have grown increasingly complex in recent years, and efficient, modular, and error-free code is a precondition for further progress (Hutton et al., 2016).

During the development of this paper we encountered several interesting – and still open – questions: The first was about where to set the system boundaries: For example, should forcing data be considered part of the model and hence be included into the counting, or not? If we consider a model that performs well even with limited input data to be more parsimonious than another, which heavily relies on information contained in the input, we should do so. But we could also argue that the input is not part of the model, and should therefore be excluded from the counting. This question also applies to the extent to which the computational setting on the computer should be included into the counting, and is open for debate. We also still struggle to provide a rigorous description of the nature and strength of the relation between descriptive and computational complexity. Clearly they describe two distinctly different characteristics of a model, but they are also related, as 'Strace' counts both the size of a program and the computational effort of running it. Like performance measured by information loss, the descriptive complexity of a model is typically orders of magnitude smaller than its computational complexity, which renders their simple additive combination to a single, overall measure of model quality impractical. Nevertheless, we suggest that combining the approach by Weijs and Ruddell (2020) with measuring computational complexity by 'Strace' will be worth exploring in the future. It potentially offers a comprehensive and multi-faceted way of model evaluation applicable across the earth sciences, where all key aspects of a model are expressed in a single unit, bit.

Code and data availability. The code and data used to conduct all analyses in this paper and the result files are publicly available at https://github.com/KIT-HYD/model-evaluation (last access: 2020/03/03).

Author contributions. EA wrote all Python scripts and code related to Strace and conducted all model runs. UE designed the study and wrote all Matlab scripts to calculate conditional entropies. EA, UE, SW, BR and RP wrote the manuscript together.

Acknowledgements. We want to thank Jörg Meyer from KIT-SCC for helpful discussions about the best way to bit-count models, Markus Götz from KIT-SCC for discussions about the LSTM model, and Clemens Mathis from Wasserwirtschaft Vorarlberg, Austria, for providing the case study data. We acknowledge support by Deutsche Forschungsgemeinschaft DFG and Open Access Publishing Fund of Karlsruhe Institute of Technology (KIT). This research contributes to the 'Catchments As Organized Systems" (CAOS) research group funded by the Deutsche Forschungsgemeinschaft DFG. RP also acknowledges FCT under references UIDB/00329/2020 and UID/EEA/50008/2019. BR acknowledges Northern Arizona University for providing start up grants used in part for this work.

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

References

Akaike, H.: A new look at the statistical model identification, IEEE Transactions on Automatic Control, 19, 716-723, 10.1109/TAC.1974.1100705, 1974.

Arkesteijn, L., and Pande, S.: On hydrological model complexity, its geometrical interpretations and prediction uncertainty, Water Resources Research, 49, 7048-7063, 10.1002/wrcr.20529, 2013.

Bennett, N. D., Croke, B. F. W., Guariso, G., Guillaume, J. H. A., Hamilton, S. H., Jakeman, A. J., Marsili-Libelli, S., Newham, L. T. H., Norton, J. P., Perrin, C., Pierce, S. A., Robson, B., Seppelt, R., Voinov, A. A., Fath, B. D., and Andreassian, V.: Characterising performance of environmental models, Environmental Modelling & Software, 40, 1-20, https://doi.org/10.1016/j.envsoft.2012.09.011, 2013.

Bras, R. L.: Complexity and organization in hydrology: A personal view, Water Resources Research, 51, 6532-6548, 10.1002/2015wr016958, 2015.

Castillo, A., Castelli, F., and Entekhabi, D.: An entropy-based measure of hydrologic complexity and its applications, Water Resources Research, 51, 5145-5160, 10.1002/2014wr016035, 2015.

Chaitin, G. J.: On the Length of Programs for Computing Finite Binary Sequences, J. Acm, 13, 547–569, 10.1145/321356.321363, 1966.

Cover, T., and Thomas, J. A.: Elements of Information Theory, Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 2006.

Engelhardt, S., Matyssek, R., and Huwe, B.: Complexity and information propagation in hydrological time series of mountain forest catchments, Eur. J. For. Res., 128, 621-631, 10.1007/s10342-009-0306-2, 2009.

Finger, D., Vis, M., Huss, M., and Seibert, J.: The value of multiple data set calibration versus model complexity for improving the performance of hydrological models in mountain catchments, Water Resources Research, 51, 1939-1958, 10.1002/2014wr015712, 2015.

Forster, K., Meon, G., Marke, T., and Strasser, U.: Effect of meteorological forcing and snow model complexity on hydrological simulations in the Sieber catchment (Harz Mountains, Germany), Hydrology and Earth System Sciences, 18, 4703-4720, 10.5194/hess-18-4703-2014, 2014.

Gan, T. Y., Dlamini, E. M., and Biftu, G. F.: Effects of model complexity and structure, data quality, and objective functions on hydrologic modeling, Journal of Hydrology, 192, 81-103, 1997.

Grünwald, P. D.: The minimum description length principle. The MIT Press, Cambridge, MA, USA, 2007.

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

Hutton, C., Wagener, T., Freer, J., Han, D., Duffy, C., and Arheimer, B.: Most computational hydrology is not reproducible, so is it really science?, Water Resources Research, 52, 7548-7555, 10.1002/2016wr019285, 2016.

Jovanovic, T., Garcia, S., Gall, H., and Mejia, A.: Complexity as a streamflow metric of hydrologic alteration, Stochastic Environmental Research and Risk Assessment, 31, 2107-2119, 10.1007/s00477-016-1315-6, 2017.

Kirchner, J. W.: Getting the right answers for the right reasons: Linking measurements, analyses, and models to advance the science of hydrology, Water Resources Research, 42, 10.1029/2005wr004362, 2006.

Kolmogorov, A. N.: Three approaches to the quantitative definition of information, International Journal of Computer Mathematics, 2, 157-168, 10.1080/00207166808803030, 1968.

Kullback, S., and Leibler, R. A.: On Information and Sufficiency, Ann. Math. Statist., 22, 79-86, 10.1214/aoms/1177729694, 1951.

Levin, D. and Syromyatnikov, E.: strace linux syscall tracer, https://strace.io, 2018.

Nash, J. E., and Sutcliffe, J. V.: River flow forecasting through conceptual models part I -- A discussion of principles, Journal of Hydrology, 10, 282-290, 1970.

Nicolis G., and Nicolis C.: Foundations of Complex Systems, 2nd edition. World Scientific Publishing Co. Pte. Ltd., Singapore, 2012.

Orth, R., Staudinger, M., Seneviratne, S. I., Seibert, J., and Zappa, M.: Does model performance improve with complexity? A case study with three hydrological models, Journal of Hydrology, 523, 147-159, 10.1016/j.jhydrol.2015.01.044, 2015.

Ossola, A., Hahs, A. K., and Livesley, S. J.: Habitat complexity influences fine scale hydrological processes and the incidence of stormwater runoff in managed urban ecosystems, J. Environ. Manage., 159, 1-10, 10.1016/j.jenvman.2015.05.002, 2015.

Ott, E.: Chaos in Dynamical Systems, 2nd edition, Cambridge University Press, Cambridge, 2002.

Pande, S., and Moayeri, M.: Hydrological Interpretation of a Statistical Measure of Basin Complexity, Water Resources Research, 54, 7403-7416, 10.1029/2018wr022675, 2018.

Perdigão, R.A.P.: Synergistic Dynamic Theory of Complex Coevolutionary Systems. https://doi.org/10.46337/mdsc.5182, 2018

Perdigão, R.A.P., Pires, C.A.L. and Hall, J.: Disentangling Nonlinear Spatiotemporal Controls on Precipitation: Dynamic Source Analysis and Predictability. https://doi.org/10.46337/mdsc.5273, 2019

Rissanen, J.: Information and complexity in statistical modeling. Springer, New York, USA, 2007.

Sang, Y. F., Wang, D., Wu, J. C., Zhu, Q. P., and Wang, L.: Wavelet-Based Analysis on the Complexity of Hydrologic Series Data under Multi-Temporal Scales, Entropy, 13, 195-210, 10.3390/e13010195, 2011.

Schoups, G., van de Giesen, N. C., and Savenije, H. H. G.: Model complexity control for hydrologic prediction, Water Resources Research, 44, 14, 10.1029/2008wr006836, 2008.

Schwarz, G.: Estimating the Dimension of a Model, Ann. Statist., 6, 461-464, 10.1214/aos/1176344136, 1978.

Sivakumar, B., and Singh, V. P.: Hydrologic system complexity and nonlinear dynamic concepts for a catchment classification framework, Hydrology and Earth System Sciences, 16, 4119-4131, 10.5194/hess-16-4119-2012, 2012.

Sivakumar, B., Jayawardena, A. W., and Li, W. K.: Hydrologic complexity and classification: a simple data reconstruction approach, Hydrol. Process., 21, 2713-2728, 10.1002/hyp.6362, 2007.

Solomonoff, R. J.: Complexity-based induction systems: Comparisons and convergence theorems, IEEE Transactions on Information Theory, 24, 422-432, 10.1109/TIT.1978.1055913, 1978.

Solomonoff, R. J.: A formal theory of inductive inference. Part I, Information and Control, 7, 1-22, https://doi.org/10.1016/S0019-9958(64)90223-2, 1964.

Weijs, S. V., and Ruddell, B. L.: Debates: Does Information Theory Provide a New Paradigm for Earth Science? Sharper Predictions Using Occam's Digital Razor, Water Resources Research, 56, e2019WR026471, 10.1029/2019wr026471, 2020.

Weijs, S. V., Schoups, G., and van de Giesen, N.: Why hydrological predictions should be evaluated using information theory, Hydrol. Earth Syst. Sci., 14, 2545-2558, 10.5194/hess-14-2545-2010, 2010.

Zhou, Y., Zhang, Q., Li, K., and Chen, X. H.: Hydrological effects of water reservoirs on hydrological processes in the East River (China) basin: complexity evaluations based on the multi-scale entropy analysis, Hydrol. Process., 26, 3253-3262, 10.1002/hyp.8406, 2012.