https://doi.org/10.5194/hess-2020-128
Preprint. Discussion started: 6 April 2020

Hydrology and Earth System Sciences
Discussions

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 parsimonious yet well-performing 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 for computer models is therefore a key theoretical and practical challenge for 21st century science. The basic dimensions of computer model parsimony are descriptive complexity, i.e. the length of the model itself, and computational complexity, i.e. the model's effort to provide output. Descriptive complexity is related to inference quality and generality, and Occam's razor advocates minimizing this complexity. Computational complexity is a practical and economic concern for limited computing resources. Both complexities measure facets of the phenomenological or natural complexity of the process or system that is being observed, analysed and modelled.

This paper presents a practical technique for measuring the computational complexity of a digital dynamical model and its performance "bit by bit". Computational complexity is measured by the average number of memory visits per simulation time step in bits, and model performance is expressed by its inverse, information loss, measured by conditional entropy of observations given the related model predictions, also in bits. We demonstrate this technique by applying it to a variety of watershed models representing a wide diversity of modelling strategies including artificial neural network, auto-regressive, simple and more advanced process-based, and both approximate and exact restatements of experimental observations. Comparing the models revealed that the auto-regressive model poses a favourable trade-off with high performance and low computational complexity, but neural networks and high-time-frequency conceptual "bucket" models pose an unfavourable trade-off with low performance and high computational complexity. We conclude that the "bit by bit" approach is a practical approach for evaluating models in terms of performance and computational complexity, both in the universal unit of bits, which also can be used to express the other main aspect of model parsimony, description length.

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). Moreover they should also contain elements of brevity, elegance, explainability, understandability, communicability, teachability, and smallness. Mathematical analytical models which are highly accurate - 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 parsimony (high elegance, brevity, and communicability). A good scientific model is parsimonious, and if it also performs well in terms of accuracy and precision it will also be useful for applied decision-making and prediction.

1.2 Occam's razor, parsimony and descriptive complexity

Occam’s Razor argues that the most parsimonious model is preferable, at a given level of predictive performance that is adequate to the question or application at hand. This bedrock principle of science allows scientists to identify preferable models as that set of models which are more parsimonious than all known and stated alternative models, at any given level of predictive performance. Therefore, 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), parsimony of a model is expressed by its descriptive complexity, i.e. its size expressed in bit, when stored as instructions for a computer. It is noteworthy that in terms of Occam's razor, the preference for models with low descriptive complexity is completely independent of any practical considerations such as limited storage space or computing power. While these may be important in practical settings, they are not of concern in applying Occam's Razor in the process of inference to lead to the model closest to the truth or giving the best predictions out of sample. In other words, the modelling is not the tool to get good compression in a storage limited world, 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 parsimony, also model performance, and to express them in the language of AIT as two additive terms of essentially the same quantity: the description length of the data in bits. 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 achieves a uniquely excellent balance of minimum model size (maximum parsimony) and minimum information loss (maximum performance), and maximum generalizability outside the observed datasets used to construct and test the model. 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 parsimony and performance: Applying a single measure, expressed in bits, to quantify both a model's performance and parsimony, 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 greatly helps guiding model preference, especially in automated environments for learning models from data.

1.3 Computational complexity

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, and are less concerned with the effort of actually applying a model, i.e. to make predictions. This effort, however, can be an important quality of a model in any setting when computing resources are limited. In earth science modelling, this is the rule rather than the exception for the following reasons, as 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, and is referred to as computational complexity, and not to be confused with descriptive complexity. In contrast to the latter, computational complexity serves the practical purpose of defining the resources needed to execute program instructions. A simple example to illustrate the relation of descriptive and computational complexity: 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.

Computational complexity 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. Although a typical personal computer clock beats about 2 billion times faster than a human heart, the time it takes for computations to finish can still be critical for our applications. 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.

https://doi.org/10.5194/hess-2020-128
Preprint. Discussion started: 6 April 2020

Underlying both descriptive and computational complexity lies the phenomenological or natural complexity of the process or system that is being observed, analysed and modelled. It might be argued that fundamental natural complexity can be quantified through metrics from the classical dynamical system theories (such as fractal dimensions, Kolmogorov-Sinai entropies and related metrics, see e.g. Nicolis and Nicolis, 2007; Ott, 2002). However, dynamical system approaches are descriptive in that they assess geometric, topologic and statistical-mechanic aspects of complexity of a description (such as phase spatial representations of the system kinematic geometry and statistical mechanics), and computational complexity in terms of the mathematical architecture underlying the dynamical systems formulation. This means that natural complexity remains elusive even in classical dynamical systems approaches, similarly to descriptive and computational complexity, which also essentially only capture aspects of the system, e.g. how it manifests through detectable observables, and not the system per se in its essence (Perdigão, 2017). In that regard, the emerging concepts and pathways in information physics (Perdigão et al., 2020) hold the potential to peer deeper into this matter. In a nutshell, descriptive and computational complexity are related, as both measure aspects of natural complexity, but the nature and strength of their relation is difficult to determine and also depends on our particular choices of representing them. Nevertheless, both provide important information about the natural phenomenon or system under consideration, and are important guidelines to build and operate models.

1.4 Scope and goals of this paper

In this paper, we take a practical and experimental approach to measure computational complexity by counting the total number of memory visits while running a model on a computer. The counting is sensitive to the size of the model and the size of the input data it reads (i.e. aspects related to storage), but also aspects related to its computational efficiency such as its numerical scheme, time-stepping and runtime environment. Additionally, we measure model performance by its inverse, information loss, by conditional entropy of observations given the related model predictions, also in bits. 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. Watersheds are complex systems that are impossible to observe or model completely and precisely, and are therefore an excellent sandbox for demonstration of this idea. Akin to Weijs and Ruddell (2020), who measured models along the dimensions of model descriptive complexity vs. information loss, we examine possible trade-offs between computational complexity vs. information loss. These trade-offs can be important in operational settings where the speed of information processing is a bottleneck.

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 used for testing, and the implementation environment and criteria for measuring model performance and computational complexity. In section three, we present and compare all 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.

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

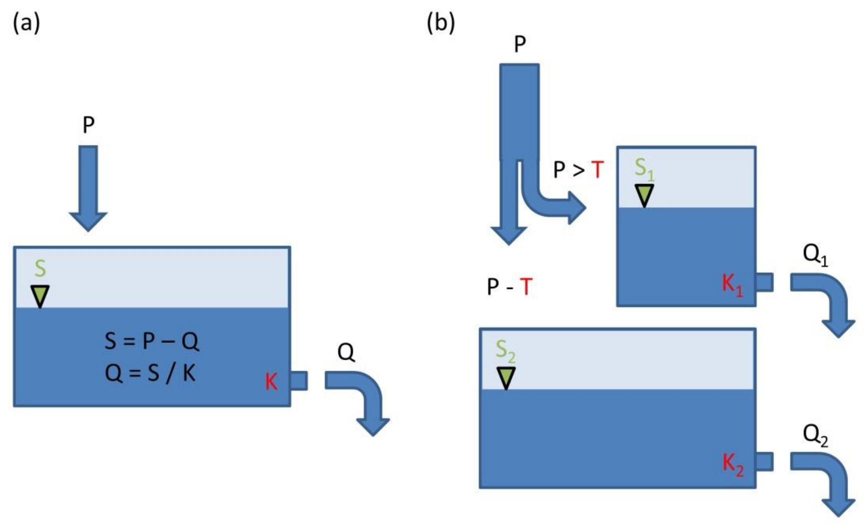
We selected altogether eight modelling approaches with the aim of covering a wide range of model characteristics such as type (ignorant, perfect, conceptual-hydrological and data-driven), structure (single and double linear reservoir), numerical scheme (explicit and iterative) and precision (double and integer). The models are listed and described in Table 1, additional information is given in Eq. 1 and Fig. 1.

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 (4.59 m³/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. 1, left panel) with a single parameter - K -, and a single state variable - S. K was found by calibration on the test data (K = 55 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	1 h	integer	explicit	Q_Host	int(P_Ebnit(*t*))

	precision only				P_Ebnit	
Model-05	A more advanced conceptual model (Fig. 1, right panel). Precipitation input is split by an intensity threshold - T - (2 mm/h), and enters two linear reservoirs - K1 - (10 h) and - K2 - (55 h). All parameters were found by calibration on the test data. 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 previous observations (see Eq. 1). All coefficients were found by calibration on the test data (c_0 = 0.0549, c_1 = 1.9266, c_2 = -1.2071, c_3 = 0.2685). Testing models of various order we found that adding observations beyond lag-3 improved predictive power only marginally	1 h	double	--	Q_Host P_Ebnit	Q_Host(*t*-3) Q_Host(*t*-2) Q_Host(*t*-1)
Model-08	A simple feedforward artificial neural network (ANN) with a single hidden layer of 10 neurons, using P_Ebnit(*t*) as input to predict Q_Host(*t*). The model is a sequential model written in Python with the "Keras" library. For the hidden layer it uses the "sigmoid" activation function, and the learning process uses the loss function "mean squared error".	1 h	double	--	Q_Host P_Ebnit	P_Ebnit(t)

$$Q_{Host}(t) = c_0 + c_1 \cdot Q_{Host}(t-1) + c_2 \cdot Q_{Host}(t-2) + c_3 \cdot Q_{Host}(t-3) \quad (1)$$



150

Figure 1. (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.

We trained/calibrated each model on the entire data. This is to maintain consistency with approaches such as advocated in Weijs and Ruddell (2020), where model performance is also evaluated on the training data and model complexity penalization would prevent overfitting. In the present paper, the purpose is not model complexity control for optimal inference, but investigating the predictive performance vs. computational efficiency trade-off. While the model's out-of-sample performance would be perhaps more interesting, we favoured consistency in objectives for demonstrating the bit-by-bit evaluation concept across a wide range of model types. Also, as we deliberately selected models with just a few calibration parameters, which is in fact a manual, non-quantified application of Occam's Razor, the risk of overfitting was low given the large number of training data (87650 time steps).

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 universality and comparability.

2.4.1 Model performance

As suggested in Weijs and Rudell (2020), we express model performance in terms of the increased uncertainty (= information loss) about the true value of the target when we have to rely on the model prediction instead of knowing the real observation. This uncertainty can be quantified by conditional entropy (see Eq. 2), where X represents the target and Y the model predictions (= predictor). Note that this assumes that our state of knowledge for each prediction is given by the observed error distribution around the deterministic prediction. 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.

To avoid fitting of theoretical functions to the empirical data distributions, we calculated conditional entropy of discrete (binned) distributions, i.e. normalized empirical histograms. We applied uniform binning by defining a value range of 0 - 150 m³/s covering all observed and simulated values of Q_Host (0.05 – 137 m³/s) and uniformly split the range into 150 bins of 1 m³/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.

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

where *X* is the target, *Y* is the predictor, *y* is a particular prediction, H(*X*|*Y*) is conditional entropy in (bit)

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

https://doi.org/10.5194/hess-2020-128
Preprint. Discussion started: 6 April 2020
© Author(s) 2020. CC BY 4.0 License.

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.

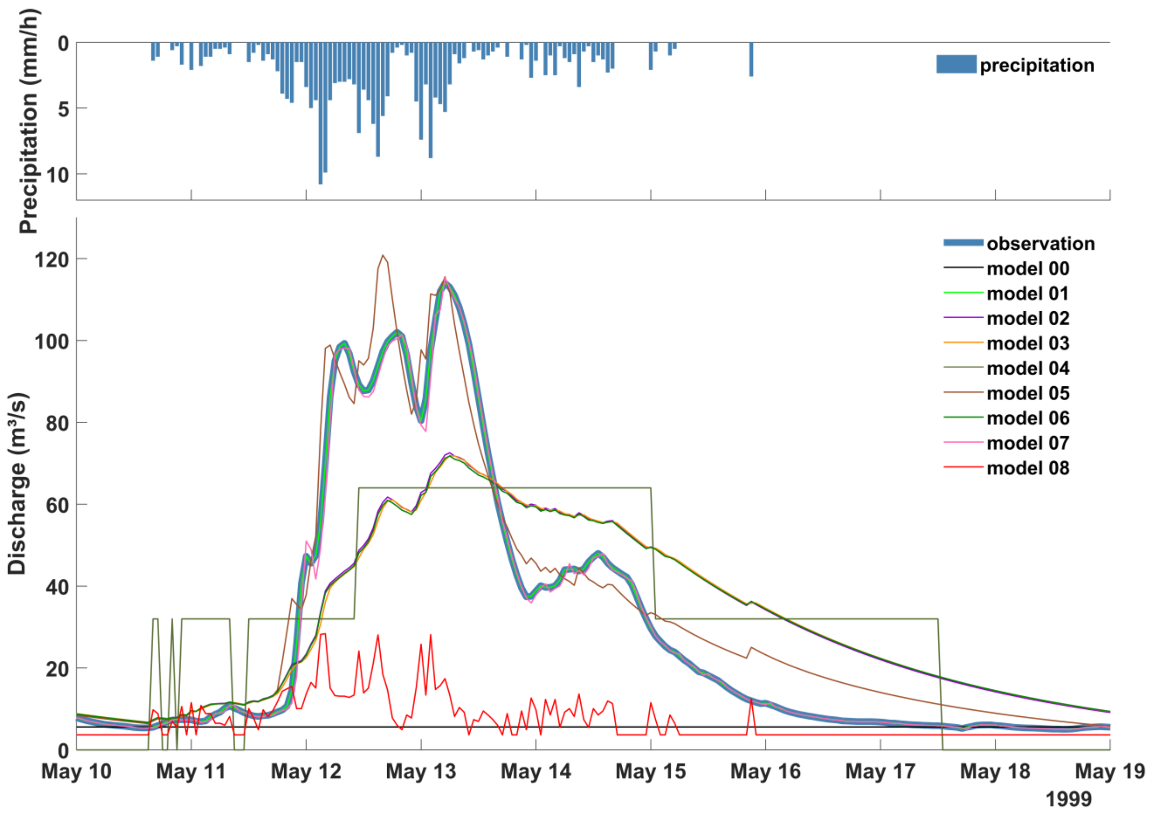
To evaluate the reproducibility of the countings, we repeated each model run 100 times, clearing the memory cache between individual runs. As the countings were in fact very close, we simply took the average of all runs as a single value representing model computational complexity. The main steps of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

3.1 Simulation vs. experimental observation

In Fig. 2, 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 May 1999. The observed hydrograph (bold blue) shows four distinct peaks caused by three larger and one lesser rainfall events embedded in a multi-day period of rainfall. 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 catchments' 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 three-step series. From all linear reservoir models, the two-bucket Model-05 (brown) performs best, correctly reproducing the overall course of the event and the embedded major peaks. The ANN Model-08 (red) clearly underestimates the event, and its dynamics are a 1:1 reflection of the observed precipitation dynamics. The former can be explained by low flow periods governing the training of the model, the latter by precipitation as the sole model input in combination with the non-recurrent nature of the ANN we used. Applying a recurrent network with the ability to capture the memory effects of the catchments' rainfall runoff transformation would most likely yield better results, but as mentioned, our goal was not to find the best-performing model, but to compare a range of different modelling approaches in terms of performance and computational complexity.



255 **Figure 2.** 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 May 1999.

https://doi.org/10.5194/hess-2020-128
Preprint. Discussion started: 6 April 2020

3.2 Performance vs. computational complexity

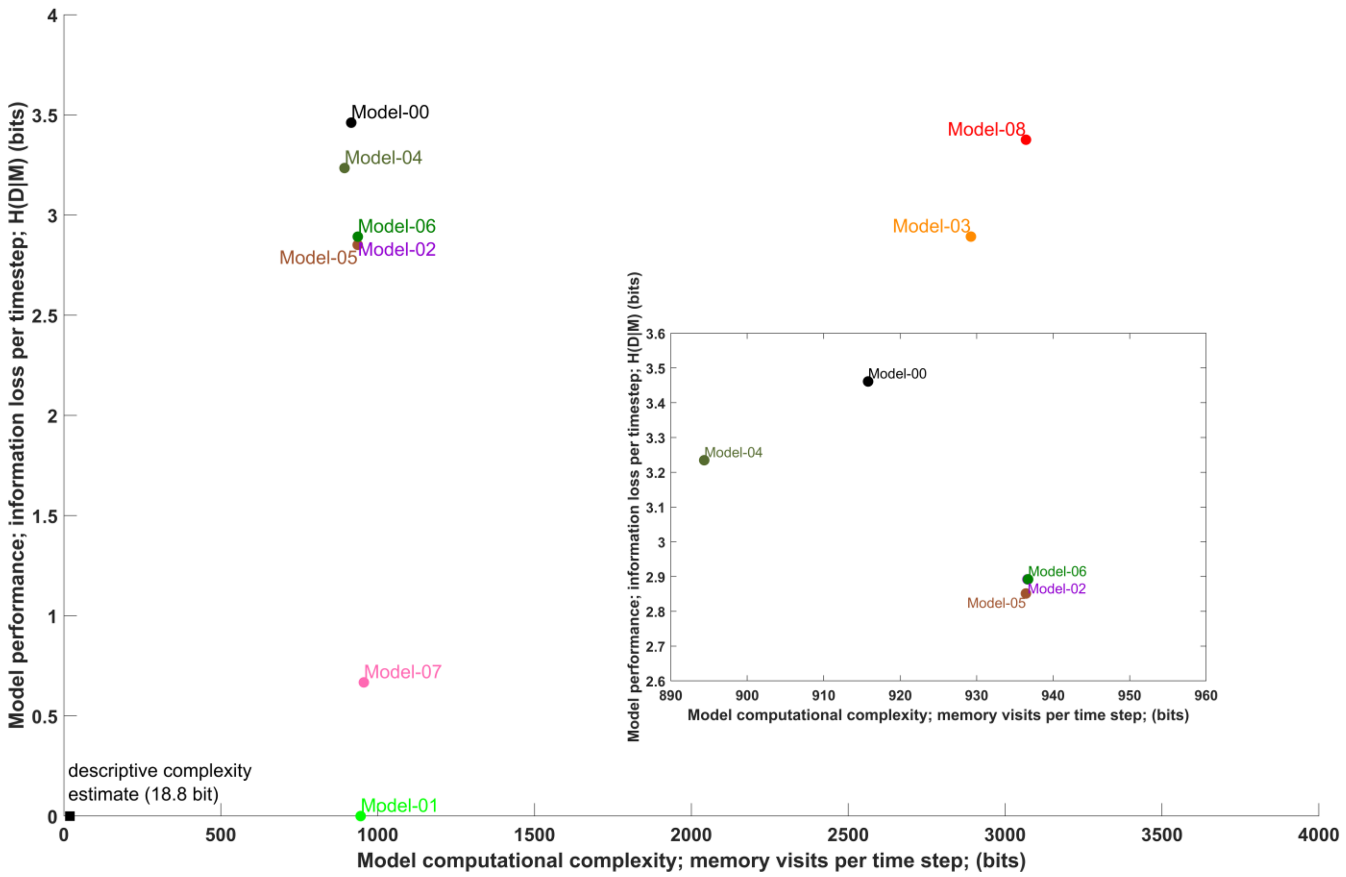
In the previous section we discussed model performance in terms of hydrologically informed visual comparison of observed and simulated hydrographs. Now we will evaluate the models in terms of both model performance and model computational complexity. Model performance is expressed as the remaining uncertainty, 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 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 simulation period (87650 time steps).

Fig. 3 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. The single-bucket Model-02 requires a higher computational effort, 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 without being rewarded by better model performance; a disadvantage not visible in Fig. 2. The poor performance of low-precision Model-04 however is visible in both Fig. 2 and Fig. 3, but the associated computational savings are 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. 2, 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 very well (second-best, only outperformed by the perfect Model-01), and with respect to computational complexity 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 neural network Model-08, on the contrary, performs poorly and, to make things worse, does so in an inefficient manner: For predicting a single time step, it visits more than 3000 bits of memory, but the remaining uncertainty about the observation still amounts to 3.37 bit.

In the lower left corner of Fig. 3, 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 dataset by the number of time steps. This represents the raw size of a single data point in the series, without any

https://doi.org/10.5194/hess-2020-128
Preprint. Discussion started: 6 April 2020
© Author(s) 2020. CC BY 4.0 License.

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.



295 **Figure 3.** 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. The inlay shows a zoom of the upper left region of the main figure.

4 Summary and conclusions

In this paper we suggested a practical technique for measuring the computational complexity and the performance of digital 300 models and provided a proof-of-concept at the example of a range of watershed models. We started by stating that one of the main objectives of the scientific enterprise is the development of parsimonious yet well-performing models for natural phenomena and systems. Occam's razor in this context is a guideline to promote parsimonious models (models with low descriptive complexity) 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. We continued by repeating Weijs and Ruddell's (2020) argument that by focusing on minimum description length only, Occam's razor achieves only "weak parsimony", and that "strong parsimony" can be achieved by adding model performance as a criterion. Weijs and Ruddell (2020) suggested that in the framework of Algorithmic Information Theory, both aspects can conveniently be expressed in bits and added to a single measure of strong model parsimony. In this paper, we maintained this information-theoretic view, but focused on a different aspect of model complexity: Computational complexity, which is the effort required by a model to generate its output. Descriptive and computational complexity both measure aspects of the natural complexity of the phenomenon or system under investigation, but the nature and strength of their relation is difficult to determine and also depends on subjective choices. Nevertheless, both provide important information about natural complexity, and both are important guidelines to build and operate models.

We measured computational complexity by counting the total number of memory visits (in bit) using the Linux tool "Strace", while running a model on a computer. The counts are sensitive to the size of the model and the input data, but also to the model's numerical scheme, time-stepping and runtime environment. We also measured model performance by the conditional entropy of observations given the model predictions. For demonstration, we ran hydrological models of various types (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 a simple artificial neural network and an unnecessarily high-time-resolution conceptual model showed very high computational complexity, which did not pay off in terms of performance. For all models, computational complexity (in bit) exceeded the missing information (in bit) expressing models 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.

To summarize, we have introduced and provided a proof-of-concept of a practical approach of measuring computational complexity and performance of computer models, both in bits, which can be used together to guide model analysis and optimization in a pareto trade-off manner, 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. An interesting question we encountered during development of our test cases was about where to set the system boundaries: For example, should forcing data be considered part of the model and 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.

Like the method proposed by Weijs and Ruddell (2020) to identify strongly parsimonious models, our "bit by bit" approach is universal in the sense that it is applicable to any model executable on computer, and it relies on a single unit, bit. Combining the strong parsimony-objective by Weijs and Ruddell (2020) with the "bit by bit" approach focusing on operational efficiency potentially yields a comprehensive and multi-faceted way of model evaluation applicable across the earth sciences.

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, 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.

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.

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.

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.

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.

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

Perdigão, R.A.P.: Fluid Dynamical Systems: from Quantum Gravitation to Thermodynamic Cosmology. M-DSC Monograph (Hard copies for order at www.fluiddynamicalsystems.com), 2017.

Perdigão, R. A. P., Ehret, U., Knuth, K. H., and Wang, J.: Debates: Does Information Theory Provide a New Paradigm for Earth Science? Emerging Concepts and Pathways of Information Physics, Water Resources Research, 56, e2019WR025270, 10.1029/2019wr025270, 2020.

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

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

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.