

PRELIMINARIES TO PRAGMATIC MODEL VALIDATION

N. BOTTA, C. IONESCU, AND R. KLEIN

1. INTRODUCTION

In spite of the preliminary output of the PIRSIQ project suggesting we do not know or we have very different opinions about what models are, we can still find it profitable to discuss a few principles and rules of model validation. This is interesting because it can lead to an increase in the quality of our models while avoiding the pitfalls of a philosophical debate. In other words, we are taking an operational approach.

We are now going to specify the framework for this discussion. We are not referring to those conceptual models that can be described in terms of a few ordinary differential equations, but rather models that heavily rely on numerical solution of systems of partial differential equations, e.g. to describe the time evolution of temperature, tracers and other model variables, and that are implemented in large codes, say of the order of 100000 lines. For example models like LM and MOM.

These models consist of different *components*. For concreteness, let's think of the components of a climate model as models for those physical subsystems which make up the climate system: vegetation, soil, ocean, atmosphere etc. Each component may, in turn, be decomposed in subcomponents. Of course, components and subcomponents can be identified according to different criteria which, in turn, lead to different classifications, model designs and implementations. The atmosphere, for instance, is usually decomposed into a *dynamics* and into a *physics*¹ the first one accounting for about one half of the code making up the whole model (e.g. in *LM*, the model used at the German meteorological office for operational numerical weather prediction).

For these kind of models the rules of model validation are far from being crystal clear. The expression model validation itself is often used together with (or in place of) expressions like model *verification*, *evaluation*, *assessment*. There is a vague impression that model validation has something to do with model correctness (haven't we sometimes heard or said that one cannot claim a model is correct if it has not been validated?).

Many activities, from business to scientific research, critically depend on component validation. Aeroplane engines are asked to go through a well defined suite of tests, both on the ground and in flight, before they get certified for operative deployment. The requirements engines have to fulfill are part of the contract between the client, the aeroplane manufacturer, and the engine manufacturer. In software engineering the need for correctness and verifiability led to programming models in which the specification of component requirements and obligations has become part of the programming language itself, see [6], [7]. In the scientific community there is a broad agreement on a few basic principles which seem to be necessary for ensuring

¹although it might seem strange – each of us has learned dynamics as a part of physics, – this classification is based on a solid argument

fairness and credibility. In the process of reviewing manuscripts – the “components” of a scientific journal – for publication, the identity of referees is usually hidden to the authors².

In the first part of this note we deduce rules of model component validation from well established principles of model inter-comparison. The implementation of such rules, however, requires criteria for selecting relevant components and for designing crucial numerical experiments and tests. In the second part we establish such criteria while embedding the practice of model validation in a more general framework. In the concluding section, we review the approach presented in order to avoid potential misunderstandings.

2. MODEL INTER-COMPARISON AND COMPONENT VALIDATION

For disciplines which cannot rely on precisely reproducible experiments like, e.g., nuclear weapon manufacturing, climate modeling and earth system analysis, model inter-comparison is a very important step of the process of accumulating evidences, investigating conjectures and identifying new paths of research. In this section we show that, in fact, simple rules of component validation can be naturally deduced from a few principles of model inter-comparison.

2.1. First principle: mandatory component validation. In some scientific communities, [2], [3], [8], [9], [4], [1], [5], [10], mandatory prior component validation has been adopted as one very important principle of model inter-comparison. Mandatory prior component validation means that researcher or a group of researchers wanting to show the capability of a model M in simulating some complex system S is not automatically entitled to do so. To qualify for a model inter-comparison, the authors of M are asked to show the ability of M in simulating some relevant subsets of S . Note that this has nothing to do with control runs or case studies. The *quality* which is required from M is not the capability to simulate different realizations of the complex system S – this is a quality to be addressed through model inter-comparison. More simply, M is required not to fail in some well defined ideal cases. These are constructed by reducing S to some of its components, say S_1 and S_2 . What is tested here are the model components M_1 , M_2 associated with the respective system components.

A practical implementation of this principle requires, of course, some agreement on the rules for defining what the ideal cases – the components – are. It also requires some theoretical knowledge because, in many cases, observations of the simplified system are not available or not meaningful. In spite of these difficulties, this principle of model inter-comparison has three remarkable properties.

First, it makes explicit a very natural requirement on M . Namely that of *consistency*. Just as in the case of a physical theory, consistency plays a fundamental role in measuring model quality. The capability of M to provide consistent representations of S through its different levels of complexity is closely related to our perception of quality. We do not feel comfortable while taxiing for take-off in a plane whose sophisticated fly by wire system has been built on the top of an old fashioned leakage prone hydraulic subsystem. We do not trust hardware floating point square root computations if the microprocessor manifestly fails in integer addition. In other words: the capability to perform easy tasks is usually considered a *necessary* pre-condition – and therefore systematically checked – for attempting to

²the symmetric rule, hiding the author’s identity to the referees, would be, of course, even more important to enforce. Unfortunately, its implementation appears to be difficult in practice

solve more difficult tasks.

Second, the component validation principle mentioned above allows for checking those properties we *know* every M *must* fulfill. In spite of the different opinions on what models are, we can expect firm agreement on a very few properties the models considered here must have. One is, for instance, invariance w.r.t. changes of units of measurements. Checking for this property does not always require a simplifications on S . Other properties following from the covariance principle - e.g., invariance with respect to symmetrical transformations – can be better tested on idealized systems.

The third property is economical. Model inter-comparison is a very expensive activity. In particular climate models are, because of the longer simulation times and the uncertainties of observational data, extremely expensive to compare. It seems natural to qualify for comparison only those models that fulfill some well known, widely accepted minimal requirements.

The clarity and openness of such requirements are an important measure of the quality standards of scientific communities whose outputs heavily rely on computer based model simulations.

2.2. Second principle: partially unknown target. Another principle of model inter-comparison states that a relevant part of the comparison should take place on partially unknown targets. In other words, the best outcome of a simulation, the one that would imply minimal distance between M and S , should not be known to the participants. Many model inter-comparison workshops in well-established disciplines of scientific computing adopted strict rules to enforce this principle, see [4], [5], [10].

The reason for the second principle is very simple: Model inter-comparison is supposed to provide some measure of the simulation capability of different models. If S is fully known to the participants, simulation capability can be imitated by optimal parameter identification. Increasing the number of *tuning* parameters of a model is a straightforward expedient to imitate simulation capability. It is, of course, a pact with the devil. For the sake of *imitating* simulation capabilities we sign a blank check. This has to be payed back in terms of exponential growth of model complexity, exponential cost of uncertainty propagation studies and exponential decrease of model predictive capability.

The second principle of model inter-comparison is essential in preventing models from imitating simulation capability through optimal identification of *extra* parameters introduced only in order to fit a data set (not germane to the problem). This principle does not prohibit their introduction. It simply makes their use worthless. Under rules instantiating the second principle of model inter-comparison, scientists are explicitly discouraged to add unjustified complexity into the models for the purpose of better fitting data. Thus, the principle re-enforces simplicity, a very important quality in science. It also guarantees both fairness and consistency: if we want to compare the capability of anti-missile systems to detect and destroy enemy targets, it would be quite stupid to give the system providers very detailed information about position, velocity and size of the targets we intend to use in the comparison!

Although indirect, the consequences of the second principle of model inter-comparison on component validation are remarkable. By discouraging the introduction of unjustified complexity, this principle makes component validation altogether possible. On the contrary, in a model relying on ad hoc tuned numerical dissipation, restoring conditions relaxation parameters and error redistribution schemes component validation is nearly impossible because extra parameters introduce artificial couplings between components.

2.3. Conclusions from model intercomparison. We have seen that a very pragmatic need – that of relying on results obtained by comparing different models and model simulations – allows us to deduce some simple rules of model component validation. It also sheds some light on the aims of model validation, one being to ensure that models qualify for model inter-comparison. Let’s now look at the issue of model validation from another point of view. This will bring a link with the concept of correctness and embed the results obtained so far in a more general framework.

3. LEVELS OF ABSTRACTION AND MODEL VALIDATION

3.1. Hierarchy of models. Let’s take a very particular view and look at a model as a computer *code*. Since the models we are talking about are usually implemented, this seems an acceptable point of view. The code is an implementation, in some computer language, e.g., FORTRAN, of the *data structures* and of the *algorithms* used to describe the *discrete* model. Thus, the components of the code are computer language maps of some abstract data structures and algorithms. Of course they are not the data structures and the algorithms themselves since these can be implemented in different languages and, in a given language, in different styles and with different degrees of correctness. Data structures and algorithms are the basic components of the discrete model. This is, in many cases, a map of some *mathematical* model: a system of algebraic equations, a partial differential equation or something else. This mathematical model is a map or, if you prefer, a model, of a system, e.g., a climate system. This system is accessible through direct and/or indirect measurements. Both the mathematical model and the measurements are related to the system by virtue of physical theories, see figure 1. These theories incorporate both fundamental principles and constitutive laws.

What is the relationship between model validation and correctness, and this chain of abstractions, from the low-level code up to the envisaged system? What we are interested in, in climate science, is understanding where the gap between simulations and measurements and observations comes from. Understanding the origin of this gap allows us to get a clearer and more complete picture of the earth climate system and to improve the predictive capability of our models. The reason we stress the term prediction is the fact that climate models are used in the process of taking decisions which unfold in the future. This means that our models are not (only) required to have descriptive capabilities – this could be obtained, with minimal gap, by means of best fit techniques – but also predictive capabilities.

In order to understand the origin of this gap it is essential to constantly have a clear view of the different levels of abstraction present in our models: from the code through the discrete model to the mathematical model and, through physical theories, to the climate system and the observable data. It is this view that, together with a notion of correctness, allows us to use the proper methods of investigating the model.

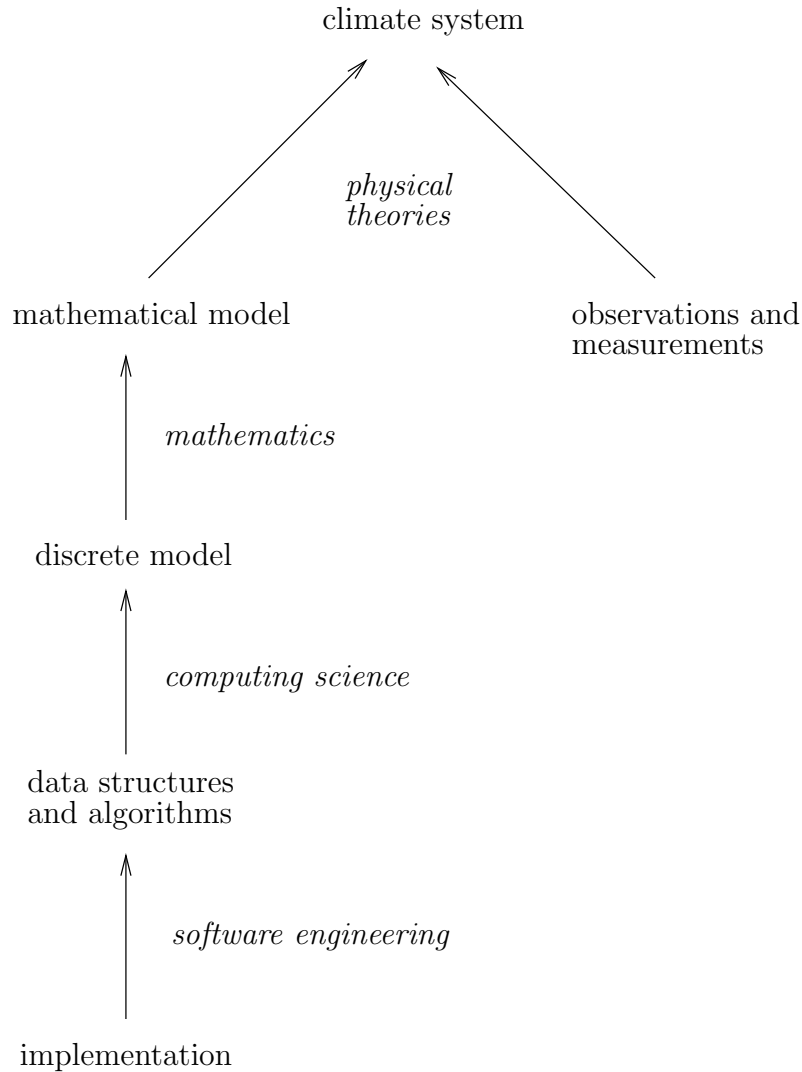


FIGURE 1. Levels of abstraction an hierarchy of models

3.2. An idealized path. Let's start, like old-fashioned mountain climbers, from the very bottom of the abstraction chain: the code. What we would like to make sure, at this level, is that our implementations are correct. If this condition is satisfied we can argue that the origin of the gap between model simulations and observations is to be found at some higher level. What does it mean for an implementation to be correct? Let's consider a simple example.

When we use `sqrt()` to compute the square root of `DOUBLE` numbers, we require this function to return, $\forall \text{DOUBLE } x \text{ such that } x \geq 0$, a result r such that $|r^2 - x| < \varepsilon$ where ε is some known system constant. These are the terms of the contract and, if they are not fulfilled, we say that `sqrt()` is not a correct implementation of $\sqrt{\quad}$ with respect to the stated contract. The example, albeit trivial, makes clear 2 important principles:

1. The question about the correctness of implementations has to be asked in the framework of some specified contract.

2. The terms of the contract are derived from the more abstract model that the implementation represents.

Of course there are other approaches to investigating the problem of implementation correctness. The one described above is a very pragmatic one. Its advantage is that it allows the inclusion of simple software validation mechanisms at run-time. This validation mechanism does not replace a proof of correctness. It nevertheless guarantees that, for the cases encountered during execution, either the contract is not violated or violations are signaled. Another advantage of this approach is that it can be easily extended to functions and subroutines which are more complex than `sqrt()`.

The approach also makes very clear why it is important, when looking at models, to keep a clear view of the different levels of abstraction: at a certain level, the terms of the contract and therefore the conditions for correctness are derived from the immediately above level, in our example the mathematical function $\sqrt{\quad}$. If we can rely on correctness of implementations – in the mountaineering parlance the base camp – we can start investigating the relationships between the discrete model, the level immediately above the implementation, and the model it represents, the mathematical model. Here the question of correctness can be stated in similar terms. What is the contract between the discrete model and the mathematical model it represents? How can we show that the discrete model fulfills the terms of the contract?

Once we have answered these questions and we know that our discrete method is a trustworthy representation of its continuous ancestor we can put the blame for the gap between model simulations and observations on the mathematical model or above. Thus, step by step, we climb the chain up to the highest level.

3.3. A pragmatic path. In real life, unfortunately, the old-fashioned mountaineering style does not always work. We have to recognize that, except for very elementary constructions, we are not able to provide proofs of correctness of our implementations. In much the same way we are, in most practical cases, unable to show that our discrete methods are faithful maps of our mathematical models. We are almost never able to estimate their accuracy and prove stability and convergence. In many cases this weakness is due to the fact that the properties of the mathematical model itself, the level of abstraction closest to the system, are very poorly understood.

It is because of this regrettable state of things that we have to provide evidence that, at least for some *relevant* configurations, our models fulfill the terms of the contract. This is, in short, what model validation is about. Far from being a proof or a guarantee that we can reliably apply the model, model validation is an act of humility and responsibility. We recognize our weakness and we try to give our results some kind of credibility.

In spite of the obvious need for model validation, its rules and practices are, as already discussed, neither crystal clear nor undisputed. The problem is that, for the validation to be credible, evidence of non-failure must be provided for a *relevant* set of model configurations. What does relevant mean? In the previous sections we have attempted to derive some rules of relevance from a pragmatic point of view. We have seen that a few fundamental principles of model inter-comparison lead naturally to some simple, understandable criteria. These criteria suggest the importance of thorough component validation and full scale validation on hidden targets.

Equipped with the concept of correctness and keeping in mind our inapplicable – but extremely useful! – rules of traditional mountain climbing we have now a useful scheme for explicitly constructing relevant sets of test problems to be used in model validation. We climb up the mountain and identify those difficult passages which seem to be risky. We design special numerical experiments to get some confidence that our components are not going to fail in these special cases. We compile protocols of our experiments which will allow us to reproduce them with other components. In this process we identify crucial checkpoints which then become mandatory tests in the validation process.

The outcome of this process is a mass of evidences which gives us some confidence that, indeed, our models might be something more than a very complicated best fit procedure. By virtue of being, down through the abstraction chain, faithful representations of some fundamental laws, they could indeed enjoy some kind of predictive capability. In areas which are heavily relying on model simulations and where crucial experiments of model falsification cannot be done or cannot be done at reasonable costs, e.g., in climate science, nuclear weapon manufacturing or astrophysics, the evidences provided by model validation are often the most important quality credentials.

4. CONCLUSIONS

There are two radical ways of misrepresenting the approach we have outlined.

The first way is to view our attempt as suggesting a straight-jacket in which all model development should fit. Those familiar with management practices in some parts of the industry might have flashing images of piles of documentation for every line of code, every small change in the grid size, every attempt to try out a different parameterization. They might wonder whether this is not the beginning of an era of hourly reports and justification of any login on the SP.

Behind these obviously exaggerated worries lies, however, a very legitimate misgiving: adopting such an approach might increase the time required to develop a model. Do we have to obtain credibility at such a price? In fact, guidelines for model validation should make the pace of development faster, not slower.

First, models are tested in any case: we have tried to provide some principles for deciding which tests matter most (those that are relevant and crucial). While experts might already know which tests to implement, having a critical justification for them will at least help newcomers to the field. The effort spent on validation can be streamlined, and our resources can be spent more efficiently.

Second, the failure of a relevant and crucial test has the chance of being informative. Such a failure should not just show that there is an error, but also give indications of its sources and even suggest avenues for improvement. This is a result of choosing our tests in a framework which respects the hierarchy of levels of abstraction. There is nothing new here, and the lessons coming from the software industry, for instance, show that proper testing improve the speed and the quality of development.

Third, and most important, the theoretical framework from which we have derived our criteria for relevant and crucial tests will hopefully lead to better communication between specialists. As the state of the art improves, it will become increasingly difficult for climate scientists to keep track of advances in numerical methods and the changes in hardware architectures and software practices. The clear separation

between the levels at which modeling takes place makes possible for scientists to interact with applied mathematicians and software developers. The search for criteria for model validation should not influence only the interfaces between programs, but should make us aware of those between people, without which the pace of development is not just slow: it stops.

The second radical misrepresentation of our attempt is to think we are on a quest for a sort of holy grail of modeling: validation as the solution to all our problems. We tried to avert this right from the start, which explains the tentativeness of this notes' title: "Preliminaries . . ." We believe that the problem of validation deserves a critical analysis, but Prolegomena do not a Kritik make, and everyone should feel free to contribute. We hope that many will.

REFERENCES

- [1] J. Vos A. Rizzi. Towards establishing credibility in CFD simulations. *AIAA Paper 96-2029*, 1996.
- [2] AIAA. Editorial policy statement on numerical accuracy and experimental uncertainty. *AIAA Journal*, 32:3, 1994.
- [3] Editorial board ASME. Journal of heat transfer editorial policy statement on numerical accuracy. *ASME Journal of heat transfer*, 116:797–798, 1994.
- [4] A. Dervieux, B. Van Leer, J. Periaux, and A. Rizzi, editors. *Numerical simulation of compressible Euler flows, Notes on Numerical Fluid Mechanics (Proceedings of the GAMM Workshop on Numerical simulation of compressible Euler flows, held at INRIA, Rocquencourt, June 10–13, 1986)*, volume 26. Vieweg Verlag, 1989.
- [5] J.-A. Désidéri. Some comments on the numerical computations of reacting flows over the double-ellipse (double ellipsoid). In *Proceedings of the Workshop on Hypersonic Flows for Reentry Problems Part I*, January 1990.
- [6] B. Meyer. *Eiffel: The Language*. Prentice Hall, 1992.
- [7] B. Meyer. *Object-Oriented software construction*. Prentice Hall, 1997.
- [8] F. White P. J. Roache, K. Ghia. Editorial policy statement on the control of numerical accuracy. *ASME Journal of fluids engineering*, 108:2, 1986.
- [9] P. J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa, Albuquerque, 1998.
- [10] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. Preprint 96-03 (SFB 359) Interdisziplinäres Zentrum für Wissenschaftliches Rechnen der Universität Heidelberg, 1996.