Discriminating accurate results in nonlinear models

Oriol Tintó Prims,
Barcelona Supercomputing Center
Barcelona, Spain
Universitat Autònoma de Barcelona
Bellaterra, Spain
oriol.tinto@bsc.es

Mario C. Acosta
Barcelona Supercomputing Center
Barcelona, Spain
mario.acosta@bsc.es

Miguel Castrillo
Barcelona Supercomputing Center
Barcelona, Spain
miguel.castrillo@bsc.es

Stella Valentina Paronuzzi Ticco
Barcelona Supercomputing Center
Barcelona, Spain
stella.paronuzzi@bsc.es

Kim Serradell
Barcelona Supercomputing Center
Barcelona, Spain
kim.serradell@bsc.es

Ana Cortés
Universitat Autònoma de Barcelona
Bellaterra, Spain
ana.cortes@uab.cat

Francisco J. Doblas-Reyes
Barcelona Supercomputing Center
Barcelona, Spain
Institució Catalana de Recerca i Estudis Avançats (ICREA)
Barcelona, Spain
francisco.doblas-reyes@bsc.es

POSTER PAPER

Abstract—Non-linear models are challenging when it is time to verify that a certain HPC optimization does not degrade the accuracy of a model. Any apparently insignificant change in the code, in the software stack, or in the HPC system used can prevent bit-to-bit reproducibility, which added to the intrinsic nonlinearities of the model can lead to differences in the results of the simulation. Being able to deduce whether the different results can be explained by the internal variability of the model can help to decide if a specific change is acceptable. This manuscript presents a method that consists in estimating the uncertainty of the model outputs by doing many almost-identical simulations slightly modifying the model inputs. The statistical information extracted from these simulations can be used to discern if the results of a given simulation are indistinguishable or instead there are significant differences. Two illustrative usage examples of the method are provided, the first one studying whether a Lorenz system model can use less numerical precision and the second one studying whether the state-of-the art ocean model NEMO can safely use certain compiler optimization flags.

Index Terms—Verification, Performance optimization, Numerical precision, Climate, Weather, NEMO4, Ocean, Nonlinear systems

I. INTRODUCTION

When trying to optimize a computer code to make it faster and/or more efficient, one requisite is to avoid degrading the accuracy of the results. In many cases this can be trivial to evaluate, but in the case of computational models that try to

simulate nonlinear systems (i.e. Navier-Stokes equations of fluid dynamics widely used in weather and climate models) this is far from being trivial. In these cases, any slight perturbation in the input data, any apparently inoffensive change in the algorithms, a change in the software stack, the specific compilation flags used or the use of a different machine can easily lead to differences in the final results. Dealing with such situations is not easy and it is an open topic in computational science [1] [2]. In many cases, trying to achieve exactly the same results is possible [3] but difficult, expensive and, above all, possibly useless. In a computational model there are many potential sources of error (model inputs, numerical methods, compiler optimization ...) and, as a consequence, the results produced by a simulation will have an inherent uncertainty. In problems that are highly conditioned by the initial conditions, such as weather prediction, minimizing the errors due to the numerical approximations or the computational implementation will have little or no effect on the accuracy if the model inputs still have uncertainties. In these cases, it is legitimate to ask ourselves if it is worth to invest resources on minimizing the model errors when this will have no impact on the accuracy of the results. Nevertheless, even accepting that any change will affect the outputs, it is still necessary to have the capacity to determine if a certain change produces differences that are within model variability of beyond.

This problem has been faced before. For example in [4] they design a method to evaluate that a code that solves

EU ESiWACE H2020 Framework Programme, Severo Ochoa programme, MINECO-Spain contract, Catalan government.

Fig. 1. Illustrative case of time-series of the error measured on a given value for each ensemble simulation is computed (gray) which allows the reader to discriminate if the error of a given case being evaluated is below the model uncertainty (green) and therefore the model is virtually the same or above (red) and therefore the changes modified what the model is simulating. In this illustration error and time units are arbitrary.

the shallow water equations is still accurate after a code optimization. However, the proposed method is not easily transferable to other problems. Some other examples [5] [6] rely on a visual inspection of the model outputs by a scientist with field expertise to assess whether the results are good enough. Finally, in other cases the verification rely on field specific advanced statistical methods [7] [8]. Therefore, none of the known solutions is as simple and flexible as the one presented in this manuscript.

To illustrate how the method can be used, two examples are provided. In the first of the two examples, we change the numerical precision used to simulate a Lorenz system and study whether the change in the precision impacts the quality of the results. In the second example, we compile the state-of-the-art ocean model NEMO with different compiler optimization flags and evaluate whether the optimization applied degrades the quality of the results.

II. METHOD

In order to establish if a certain change in a computational model (a different numerical method, different compilation flags, a different platform,...) changes the results beyond what can be acceptable we have to start defining a test case. A test case usually will be composed by specific model inputs (initial conditions, simulation parameters, simulation length,...), the software (source code, compiler, software stack) and the hardware (a specific platform).

With all these elements settled, we compute the test case and define the outputs obtained as our reference outputs. Having the reference outputs, we can then evaluate the differences between any given simulation and our test case. To achieve this we use the root mean square error (RMSE), whose definition can be found in equation 1, generating a time-series of the error for each variable being evaluated.

$$RMSE(t) = \sqrt{\frac{\sum_{i=1}^{i_{max}} \dots \sum_{n=1}^{n_{max}} (ref_{i,...,n}(t) - x_{i,...,n}(t))^2}{i_{max} \cdot \dots \cdot n_{max}}} \quad (1)$$

Where $i, \dots, n$ are the dimension indices.

At this point, we are only able to distinguish whether the results are exactly the same and the RMSE for all the evaluated variables is exactly 0 or if the results present some difference. In case of having a difference, we are still not able to explain if this difference is relevant or not.

In order to have something to compare with, we propose to produce a series of simulations that are almost identical to the reference case, but where the initial conditions have been slightly modified. This is widely known in Earth sciences as an ensemble of simulations. In these cases, the simulation parameters, the source-code, and everything but the initial conditions of the simulation must be exactly like the reference case. For each one of those simulations and for each output variable being evaluated we produce the RMSE time-series. At this point, by plotting the time-series we should be able to see how the RMSE of each case follows a different trajectory due to the non-linearity of the model.

With all this done, in order to evaluate the impact that a certain change has in the outputs, we just need to launch the simulation with the specific conditions that we want to test, compute the RMSE of each variable against the reference simulation and compare these RMSE time-series with the time-series of the ensemble simulations. If the time-series of the case being evaluated are similar or below the ensemble, we consider that the results are good enough, and if on the contrary the time-series show that the RMSE grows faster than the members of the ensemble, we consider that the results are not good. This is illustrated in Figure 1

Fig. 2. Numerical integration of the Lorenz system described in equation 2. In the top part of the figure, the dashed black line represents the reference solution, the gray lines represent members of the ensemble, the red line represents a case using 13-bits in the significant and the green line represents a case computed using 23-bits in the significant. In the bottom, the logarithm of the error in x between the reference and the ensemble members (gray), the 13-bit case (red) and the 23-bit case (green).

III. EXAMPLES

A. Adjusting numerical precision for a Lorenz System simulation

A Lorenz System is a simple system of equations (see equation 2) that shows chaotic behaviour. This implies that that the slightest perturbation in the system (inputs, numerical methods, ...) would make that the results of two very similar simulations end up completely different.

$$\frac{\partial x}{\partial t} = \sigma(y - x), \frac{\partial y}{\partial t} = x(\rho - z) - y, \frac{\partial z}{\partial t} = x \cdot y + \beta z \quad (2)$$

To study if the precision can be reduced without degrading the quality of the results we apply the method explained in the previous section. We perform a simulation using the reference code in double precision with parameters $\sigma = 10$, $\rho = 28$ and $\beta = 3/8$, with the initial conditions being $x = y = z = 1$. After that, we launch one hundred simulations with the same code but adding a slight perturbation ($p \propto 10^{-4}$) to the initial conditions. In Figure 2 it can be seen how the system behaves chaotically and the members of the ensemble that have very similar initial conditions (grey lines) have very different trajectories after 15 seconds.

Using a reduced precision emulator [5] that allows us to simulate what would happen if the simulations were performed using floating-point representations with a specific number of significant bits, we performed tests with different cases and evaluated them against the ensemble. The results that can be seen in Figure 2. In the top part of the figure it can be seen that using a 13-bit significant floating point representation (red line) the resulting simulation is significantly different from the reference simulation (dashed black line), far beyond the variability measured by the ensemble, while the

simulation performed with a 23-bit significant floating point representation is closer to the reference than the ensemble members. In the bottom part of figure it can be seen that with a simple visual inspection of the two reduced-precision cases against the ensemble one can determine if the approximation degrades the results beyond model uncertainty (like the red case) or the error introduced is negligible compared to the model uncertainty due to other error sources (green case).

B. Trying different compiler flags in NEMO4

NEMO is a state-of-the-art ocean model and it is based on the Navier-Stokes nonlinear partial differential equations. In order to use the NEMO source-code to perform simulations on a supercomputer we have to rely on compilers that build the actual executable. Compilers can use approximations to obtain faster programs, and some of these approximations can affect the results of the floating point arithmetic. In order to determine if the usage of an specific compiler optimization degrades the accuracy of the model, we use the method explained in this manuscript. Our test case is a one year simulation with an ORCA2 grid (two degree horizontal resolution) with which we perform a one year simulation without compiler optimization. Afterwards we launch an ensemble of simulations with the same version of the executable but starting with perturbed initial conditions (the temperature field is modified with white noise), and measure how much the ensemble members differ from the reference in the many variables that the model can output. We use this information to evaluate the effects of using certain flags with the Intel compiler on the Marenostrum 4 supercomputer hosted by the Barcelona Supercomputing Center. Modern compilers have a very extensive list of possible optimization methods that can be activated or deactivated

Fig. 3. A) Time evolution of the root mean square error of the 3D temperature field. B) Time evolution of the error in the global sea surface height.

through a compilation flag. Understanding the specific action of each flag goes beyond the scope of this manuscript, which only aims to illustrate the potential usage of the method. To do so we tested the most usual optimization flags (O1, O2, O3 and xHost).

In the results shown in Figure 3 it can be seen that while it is true that compiler optimization leads to different results, these differences are not always indicating wrong results. In Figure 3 A it can be seen that while compiler optimization prevents bit-to-bit reproducibility, the error remains smaller than the error observed in the ensemble. In Figure 3 B, it can be seen that while the flags **O1**, **O2** and **O3** produce results that are comparable to those generated without optimization, the flag **xHost**, which enables the use architecture specific instructions, leads to results that clearly differ from the ensemble for the global sum of the sea surface height.

IV. DISCUSSION

The method presented in this manuscript allows us to discriminate with a fairly solid basis whether a model modification degrades the results or these remain within its intrinsic uncertainty. It has been successfully used in two very different exercises, proving its usefulness. The Lorenz system example illustrated how by simple visual inspection it can be seen if the results are degraded beyond model uncertainty or not. The NEMO example showed that while compiler optimization leads to different results, these results remained within model variability except for the use of -xHost optimization which caused the degradation of the results for the global sea surface height. While there is room for more theoretical considerations and improvements have to be made in order to generalize even more this method, it can be a valuable resource to validate results from non-linear models. It can be specially useful for computer engineers trying to optimize the computational performance of models with non-linear behaviour and who do not have a strong background on the scientific field.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU ESIWACE2 H2020 Framework Programme under grant agreement no. 823988, from the Severo Ochoa (SEV-2011-00067) programme of the Spanish Government, from MINECO-Spain under contract TIN2017-84553-C2-1-R, and from the Catalan government under grant 2017-SGR-313.

REFERENCES

- [1] C. Collberg and T. A. Proebsting, "Repeatability in computer systems research," *Commun. ACM*, vol. 59, no. 3, pp. 62–69, Feb. 2016. [Online]. Available: <http://doi.acm.org/10.1145/2812803>
- [2] P. Ivie and D. Thain, "Reproducibility in scientific computing," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 63:1–63:36, Jul. 2018. [Online]. Available: <http://doi.acm.org/10.1145/3186266>
- [3] R. Li, L. Liu, G. Yang, C. Zhang, and B. Wang, "Bitwise identical compiling setup: prospective for reproducibility and reliability of earth system modeling," *Geoscientific Model Development*, vol. 9, no. 2, pp. 731–748, 2016. [Online]. Available: <https://www.geosci-model-dev.net/9/731/2016/>
- [4] J. S. Targett, P. Düben, and W. Luk, "Validating optimisations for chaotic simulations," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–4.
- [5] A. Dawson and P. D. Düben, "rpe v5: an emulator for reduced floating-point precision in large numerical simulations," *Geoscientific Model Development*, vol. 10, no. 6, pp. 2221–2230, 2017. [Online]. Available: <https://www.geosci-model-dev.net/10/2221/2017/>
- [6] P. D. Düben and T. N. Palmer, "Benchmark tests for numerical weather forecasts on inexact hardware," *Monthly Weather Review*, vol. 142, no. 10, pp. 3809–3829, 2014. [Online]. Available: <https://doi.org/10.1175/MWR-D-14-00110.1>
- [7] F. Váña, P. Düben, S. Lang, T. Palmer, M. Leutbecher, D. Salmond, and G. Carver, "Single precision in weather forecasting models: An evaluation with the ifs," *Monthly Weather Review*, vol. 145, no. 2, pp. 495–502, 2017. [Online]. Available: <https://doi.org/10.1175/MWR-D-16-0228.1>
- [8] T. Thornes, P. Düben, and T. Palmer, "On the use of scale-dependent precision in earth system modelling," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 897–908, 2017. [Online]. Available: <https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.2974>