Report on the 2023 Workshop on Correctness and Reproducibility for Climate and Weather Software

Alper Altuntas
Allison Baker
Ilene Carpenter
Brian Dobbins
Michael Duda
Dorit Hammerling
Thomas Hauser
Karsten Peters-von Gehlen

NCAR Technical Notes
NCAR/TN-582+PROC

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

NCAR TECHNICAL NOTES

http://library.ucar.edu/research/publish-technote

The Technical Notes series provides an outlet for a variety of NCAR Manuscripts that contribute in specialized ways to the body of scientific knowledge but that are not yet at a point of a formal journal, monograph or book publication. Reports in this series are issued by the NCAR scientific divisions, serviced by OpenSky and operated through the NCAR Library. Designation symbols for the series include:

EDD – Engineering, Design, or Development Reports
Equipment descriptions, test results, instrumentation, and operating and maintenance manuals.

IA – Instructional Aids
Instruction manuals, bibliographies, film supplements, and other research or instructional aids.

PPR – Program Progress Reports
Field program reports, interim and working reports, survey reports, and plans for experiments.

PROC – Proceedings
Documentation or symposia, colloquia, conferences, workshops, and lectures. (Distribution maybe limited to attendees).

STR – Scientific and Technical Reports
Data compilations, theoretical and numerical investigations, and experimental results.

The National Center for Atmospheric Research (NCAR) is operated by the nonprofit University Corporation for Atmospheric Research (UCAR) under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

National Center for Atmospheric Research
P. O. Box 3000
Boulder, Colorado 80307-3000

NCAR/TN-582+PROC

NCAR Technical Note

2024-07

Report on the 2023 Workshop on Correctness and Reproducibility for Climate and Weather Software

Alper Altuntas

NSF National Center for Atmospheric Research (NCAR), Boulder, CO, USA

Allison Baker
NSF National Center for Atmospheric Research (NCAR), Boulder, CO, USA

Ilene Carpenter
Hewlett Packard Enterprise, USA

Brian Dobbins
NSF National Center for Atmospheric Research (NCAR), Boulder, CO, USA

Michael Duda
NSF National Center for Atmospheric Research (NCAR), Boulder, CO, USA

Dorit Hammerling
Colorado School of Mines, Golden, CO, USA

Thomas Hauser
NSF National Center for Atmospheric Research (NCAR), Boulder, CO, USA

Karsten Peters-von Gehlen
Deutsches Klimarechenzentrum GmbH (DKRZ), Hamburg, Germany

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
P. O. Box 3000
BOULDER, COLORADO 80307-3000
ISSN Print Edition 2153-2397
ISSN Electronic Edition 2153-2400

How to Cite this Document:

Altuntas, Alper, Allison Baker, Ilene Carpenter, Brian Dobbins, Michael Duda, Dorit Hammerling, Thomas Hauser, and Karsten Peters-von Gehlen. (2024). Report on the 2023 Workshop on Correctness and Reproducibility for Climate and Weather Software. (No. NCAR/TN-582+PROC). doi:10.5065/0534-mc88

Report on the 2023 Workshop on Correctness and Reproducibility for Climate and Weather Software

Alper Altuntas[1], Allison Baker[1], Ilene Carpenter[2], Brian Dobbins[1], Michael Duda[1], Dorit Hammerling[3], Thomas Hauser[1], and Karsten Peters-von Gehlen[4]

[1]National Center for Atmospheric Research, Boulder, CO, USA
[2]Hewlett Packard Enterprise, USA
[3]Colorado School of Mines, Golden, CO, USA
[4]Deutsches Klimarechenzentrum GmbH (DKRZ), Hamburg, Germany

June 2024

Contents

1 Preface

Model simulations are essential tools for understanding weather and climate. As we adapt to our changing climate, simulation codes inform both our understanding and policy decisions. These complex software artifacts are often the result of multiple decades of development. And they are in a state of near-constant development as scientific capabilities advance and high-performance computing (HPC) technologies evolve.

Given the societal and environmental importance of these codes, maintaining confidence and preserving code quality is critical. Yet scientific computing applications are often developed without the use of extensive software verification tools and techniques. Instead, development practices are typically dominated by short-term concerns about performance, resources, and project timelines. Technical challenges in running and evaluating climate and weather models further complicate code verification efforts. Given the scale of these models, a thorough correctness evaluation may be prohibitively expensive. Additionally, the customary requirement for regression tests to yield bitwise reproducible results is often unattainable due to the chaotic nature of these models and the diverse hardware and software environments in which they operate. When bitwise reproducibility cannot be sought, field experts are to evaluate model results in a time-consuming and subjective manner. Therefore, rigorous yet practical approaches tailored to such specific challenges of climate and weather modeling must be devised and employed to ensure robustness, accuracy, and reproducibility.

In pursuit of this goal, a promising technique known as ensemble consistency testing (ECT) has been developed and is being increasingly utilized [15, 43]. ECT is a statistical approach designed to evaluate whether alterations in software or hardware produce consistently different model outputs, possibly introducing artifacts that could influence scientific interpretations. Another technique that could enhance model accuracy is the adoption of domain-specific languages (DSLs), which facilitate the separation of concerns and consequently promote reliability through maintainability. While ECT and DSLs serve as noteworthy examples of techniques tailored to the specific requirements of climate and weather modeling, verification methods from more traditional software engineering practices can also be more widely adopted and potentially customized to suit the needs of climate and weather modeling, thereby contributing to achieving correctness and reproducibility.

For instance, traditional software testing and reviewing methods, including unit testing, regression testing, and peer reviewing, among others, are pivotal in ensuring the reliability, correctness, and robustness of software systems in industry. Therefore, these should also be systematically, to a much larger extent than currently done, utilized in research software development. Similarly, though less explored and utilized, formal methods, i.e., mathematically based rigorous techniques to specify and verify software, have shown to be effective in various case studies in industry [19, 36, 44]. Hence, whenever feasible, they may be utilized as a complementary means to improve correctness of climate and weather modeling applications. As for reproducibility, several notable techniques including version control, containerization, package management, and continuous integration, can be more extensively adopted in research software development and HPC community.

Concerns regarding correctness and reproducibility in software developed for scientific research are not uncommon (e.g., see [25, 50]). For example, these concerns arise partly due to the absence of well-established definitions for correctness and reproducibility in climate and weather modeling, along with classifications outlining different levels of these concepts. Clearly defined definitions can facilitate the establishment and fulfillment of expectations. As an initial step, we differentiate between "software correctness", ensuring the absence of bugs and precise behavior as specified, and "scientific correctness", indicating the model's fidelity to physical phenomena. Reproducibility, on the other hand, encompasses a range of objectives, from bitwise reproducibility to achieving results that are non-identical and yet do not alter the climate or weather of a simulation.

In response to the challenges, uncertainties, and opportunities outlined thus far, a workshop on "Correctness and Reproducibility for Climate and Weather Software" was held on November 9-10, 2023, at the National Center for Atmospheric Research in Boulder, CO. This collaborative event brought together the

NCAR applications community with the broader HPC community, including potential industry partners, to address the practical challenges faced by climate and weather model applications regarding correctness and reproducibility. The purpose of this report is to summarize the workshop and plan a path forward for future work, collaboration, and potential funding sources to take steps and advance towards the goals identified in the workshop. To this end, we discuss the motivation for the workshop, summarize presentations and discussions from the workshop sessions, and identify the critical issues and key takeaways, providing a roadmap for future work and collaboration.

2 Workshop Organization

The Workshop on Correctness and Reproducibility for Climate and Weather Software took place on November 9-10, 2023, at the National Center for Atmospheric Research in Boulder, CO. Co-organized by Allison Baker and Alper Altuntas, the workshop received funding from the UCAR President's Strategic Initiative Fund (PSIF) and was co-sponsored by both the Computational and Information Systems Laboratory (CISL) and the Climate and Global Dynamics Laboratory (CGD) at NCAR. The organizing committee members authored this report.

Goal. The primary goal was to raise awareness and establish collaborative partnerships to create mutually beneficial solutions for modeling applications with substantial scientific, operational, and social impact. This involved identifying challenges and existing limitations, engaging in discussions about current and future solutions, and exploring the definitions and implications of correctness and reproducibility in the context of climate and weather modeling.

Call for abstracts. Contributions to the workshop were solicited within the context of climate and weather simulation codes for the following topics of interest:

- Tools and approaches for software testing, debugging, quality assurance, and continuous integration.
- Statistical and ensemble-based approaches for evaluating consistency and software correctness.
- Approaches and development practices for streamlining correctness and reproducibility efforts.
- Formal methods, abstraction, and logical proof techniques for rigorous verification.
- Verifying and validating large-scale applications running on HPC clusters, GPUs, cloud, etc.
- Other software correctness and reproducibility approaches for facilitating verification and validation.

Contributions on the above topics related to climate and weather simulation codes could include technical results, approaches, experiences, and opinions. Specific areas of climate and weather applications include, for example, drivers, couplers, frameworks, model components, artificial intelligence techniques, diagnostics, post-processing, visualization tools, libraries, packaging, environment management, version control, and porting techniques. Though any other software development and approaches extensively used within the climate and weather simulation context were welcomed.

Review process and attendance. We received 23 abstracts, and 17 of them were accepted through a peer review process, taking into account their significance, relevance, originality, and quality, alongside time constraints. Our hybrid workshop attracted a total of 92 participants from academia, laboratories, and industry, of which 41 participated in person. Of the total participants, 22 were international attendees.

Format. The format of the workshop, which extended over two days, included four invited keynote talks, seventeen contributed talks, a panel discussion, and opportunities for community interaction during open discussion periods and networking breaks. Despite the two-day format, we note that interest in the workshop exceeded our expectations, and we were not able to grant every abstract submission a contributed talk due to time constraints.

More detailed workshop program and speaker information is included in the appendix. Further details about the workshop can be found on the workshop website at `https://ncar.github.io/correctness-workshop/`.

3 Motivation

Climate and weather models are the result of multiple years or even decades of development, resulting in large and complex code bases that continually evolve. These sophisticated software artifacts are often in a state of near-constant development, as new scientific capabilities and processes are added and ever-evolving HPC technologies require code adaptations and further optimizations. Frequent and systematic software verification is essential throughout the development life cycle to ensure that modifications to the hardware and/or software environment do not inadvertently impact the model results. Institutions acting in the field of weather and climate service and research, such as NCAR, NOAA, NASA, MPI-M and ECMWF consider maintaining confidence in model accuracy and preserving code quality and reliability as crucial, given the societal and environmental importance of these codes in navigating our changing climate. Indeed, the outputs of climate and weather model simulations play a significant role in shaping our understanding and informing policy decisions across various domains, including agriculture, public health, and responses to extreme weather events [26].

Scientific software is often developed without extensive utilization of software verification tools and techniques, particularly for the large and complex codebases whose development is largely done by domain scientists who are focused on scientific advancement [50]. Climate and weather codes certainly fall into this category, which is characterized by the popularity of bit-comparison tests as a technique for regression testing as well as an understandable scientific concern for reproducibility of experiments [25]. Due to the extensive time required for full runs and the multitude of model configurations, exhaustive and automated testing is often impractical or prohibitively expensive [21], [47]. In fact, climate scientists often evaluate code changes in a time-consuming and subjective manner that involves comparing new experimental results with control runs and observational data [25]. Further, ongoing development means that evaluating and ensuring correctness for weather and climate models is a continual challenge.

Testing a large software system extensively for correctness is a well-known challenge. For instance, when integrating a new feature without an established test oracle, distinguishing between correct and potentially incorrect behavior becomes difficult. While comprehensive, automated, and rigorous verification techniques are preferred, the only practical option in many cases is subjective and human-based, offering much less assurance. The prevalent reliance on bit-reproducible results in scientific codes exacerbates the existing challenges even further. The chaotic nature of climate and weather models means that a roundoff-level perturbation added to an initial condition or intermediate result can lead to sizable differences in the final result. This sensitivity is commonly referred to as a surrogate for the "natural variability" reflected in the model simulation. Therefore, even making a small change to the model code or its build and runtime environment, such as upgrading a machine's compiler version or changing the order of operations in a single function, acts as a perturbation and may not result in bit-identical results. Internationally established and widely applied climate and weather codes such as CESM [13], E3SM [38], UFS [12], ICON [35], IFS [6], MOM6 [9], and NEMO [11] are frequently ported to new machines, which generally involves changes in compilers, optimization levels, arithmetic and parallel libraries, and many other aspects that determine the code execution. Further, heterogeneous machines, including multicore CPUs, GPUs, and other accelerators, as well as advanced compilation methods, ensure that the output on the new machine will not be bit-identical to the original [14]. In the absence of bit-identical results,

the challenge is in determining whether the difference in simulation results is due to an error or simply to the model's natural variability. Addressing this question is somewhat tractable when the difference (e.g., a compiler upgrade) is not expected to change the scientific conclusions. However many modifications in the software lifecycle, such as adding a new physical process, are expected to produce "answer-changing" results. How do we evaluate the correctness of such modifications?

While the issue of correctness and (non-) reproducibility have received attention by computer science researchers, most of the tools and technologies developed thus far are not yet suited for large and complex HPC codes [29]. Indeed, the HPC application community, and climate and weather modelers in particular, are in need of practical and feasible means of addressing and ensuring correctness [14]. Several scientific applications, such as those relevant to NCAR's mission, are already GPU-enabled or are in the process of being optimized, which means the task of evaluating whether the new output is "correct" is quite timely and critical.

The increasing popularity of ML (machine-learning) also presents challenges to evaluating correctness and ensuring reproducibility, particularly as ML becomes more prevalent in climate modeling and weather prediction. While conventional models will not be entirely replaced by machine learning models, some believe that most weather predictions will come from machine learning models in the next several years (e.g., see [20]).

In short, climate and weather modeling communities are in need of practical and feasible means of ensuring correctness and reproducibility. For example, we are interested in means to easily assess whether changes to a model code result in output that is systematically different or introduce artifacts that could influence scientific conclusions. Such changes may include hardware or software stack infrastructure differences, replacing parts of the model with ML-routines, or applying data compression to the output data.

4 Defining Correctness and Reproducibility

In the previous sections, we mention a number of previous works that discuss correctness and reproducibility in the broader model simulation context (e.g., see [29, 28, 50]), as well as early discussions in the context of climate models (e.g., see [25, 21]). Building from these, a primary objective of this workshop has been to clarify the definitions of correctness and reproducibility within the current context of climate and weather modeling. Although sometimes used interchangeably, it's important to recognize that correctness doesn't inherently imply reproducibility, and vice versa. In essence, while reproducibility ensures consistent outcomes, it doesn't guarantee accuracy in representing physical phenomena. Similarly, achieving software correctness doesn't automatically ensure reproducible results. The lack of well-established definitions for these terms, along with classifications delineating various levels, poses challenges towards advancing scientific understanding, fostering collaboration, and improving the reliability of models. Clear and widely accepted definitions can simplify the process of establishing and meeting expectations for accurate climate and weather model simulations.

4.1 Correctness:

To begin, we distinguish between scientific correctness and software correctness. Scientific correctness relates to the faithfulness of domain-specific components, such as physics, parameterizations, grid resolutions, numerical schemes, and observational datasets, in representing reality. Software correctness, on the other hand, concerns the absence of bugs and ensures programmatic behavior precisely as specified. While our primary focus is on software correctness, it's worth noting that methods for evaluating software correctness, such as ECT, can also shed light on scientific correctness. These two forms of correctness are frequently interconnected; for example, the convergence of a numerical scheme is heavily influenced by the execution of floating-point operations and compiler optimizations, as well as truncation errors resulting

from numerical approximations of differential equations. Ultimately, the overall accuracy of the model can be seen as the combined result of scientific and software correctness.

Model accuracy =

Scientific correctness, e.g., physics, parameterizations, grid resolution, numerics, observational datasets, etc.

× **Software correctness**, e.g., algorithms, floating-point precision and operations, parallelism (e.g., lack of data races and deadlocks), compiler optimizations (e.g., fused multiply-add), software and hardware environments, etc.

4.2 Reproducibility

Reproducibility in climate and weather models encompasses a spectrum of objectives, ranging from achieving bitwise identical results to ensuring statistically consistent simulations without altering the overall climate behavior. We outline various reproducibility objectives with differing levels of stringency:

Bit-for-bit reproducibility: This is the most stringent level, aiming for bitwise identical results across different platforms, including various compilers, optimization levels, computer systems, and even different types of processors (CPUs and GPUs).

Layout reproducibility: This level ensures that, with all other factors held constant, changing the processor count or layout (i.e., how the problem is distributed among the processors) does not alter the results.

Restart reproducibility: This level guarantees that restarting a simulation from an intermediate state yields the same results as running it from the initial state.

Absence of non-determinism: This level focuses on reproducibility within the same hardware and software environment. It ensures that running the same model configuration multiple times on the same system produces identical results, confirming the absence of non-deterministic factors due to randomness or parallel processing that could introduce variations.

Statistical reproducibility: This level aims for statistical consistency in climate and weather simulations when bitwise identical results cannot be sought. Climate reproducibility ensures the overall statistical properties of the model outputs are consistent across different runs.

5 Achieving Correctness and Reproducibility

In this section, we present an overview of the diverse range of methods and tools explored during the workshop, all aimed at bolstering the correctness and reproducibility of weather and climate software. These include statistical approaches and tools, testing methodologies for bug detection, development practices promoting reliability and maintainability, analysis and diagnostics tools, benchmarks, domain-specific languages, and the application of formal methods.

5.1 General talks

Our climate is changing rapidly, and better climate models are essential to improving our understanding. Near continuous development is a hallmark of most modern ESM codes, and several presenters at the workshop gave talks that spanned multiple topics related to correctness and reproducibility needed to maintain confidence in these rapidly evolving model codes. We discuss these more general and motivating presentations first.

Steve Easterbook of the University of Toronto is well known for his early work on climate software and correctness issues [25, 26] as well as his recently released book [24] that provides an insight as to how climate models work and what we can learn from them. Easterbrook opened the workshop with an engaging keynote talk that discussed what the concept of correctness might mean for a computational climate and weather model, as well as the difficulties in defining such a concept for these models. In particular, because a model is a simplification of a physical system, it will not be perfect, as captured by the well known aphorism "All models are wrong but some are useful". Therefore, different perspectives on how to best assess model quality and correctness are common in practice, exemplified by questions such as "Is the model well-constructed?" or "Are there bugs?" or "Do model results match observations?" or "Can results be replicated?". Easterbrook argues that an alternate and more useful view of correctness requires defining the "scope of applicability" for a model, meaning to determine what aspects of the physical system that a particular model represents well. Looking at correctness from this perspective allows for models to confidently explain and predict within their scope of applicability while also acknowledging model errors and continuing to make improvements.

Peter Dueben of the European Centre for Medium Range Weather Forecasts described the state of current Earth System Modelling for weather and climate as "experiencing disruptive changes offering great opportunities." Such disruptive changes include the move toward increased kilometer-scale resolution. Increases in resolution are critical for storm-resolving atmospheres, eddy-resolving oceans, and better matching satellite data, for example, and must take advantage of increasingly heterogeneous hardware in high-performance computing (see, e.g., [16, 17]). Another rapid change is the introduction of machine learning (ML) into the climate and weather modeling domain [20]. ML has the potential to speed up predictions and/or improve physics-based models, and some research goes as far to suggest replacing conventional models, particularly for weather forecasting. The factors driving the rapid changes in Earth System Modelling underscore the need for robust correctness and reproducibilty capabilities for moving forward. Indeed, more automated correctness tools, publicly available model and observation data, and collaboration is needed to move forward.

Michael Coblenz of UC San Diego addressed the question of how can we help people who write software be more effective. In particular, many scientists create and maintain complex software artifacts in their work, but do not have formal background in software development. This situation is certainly common in weather and climate modeling, and is important as their ability to construct useful models and arrive at correct conclusions depends on their efficacy in writing software. Therefore, an important factor in improving software correctness is to provide and take advantage of opportunities to adapt software engineering tools and processes with the aim of making scientists more effective at writing software (see [46]).

5.2 Statistical Approaches

The keynote talk by Dorit Hammerling of Colorado School of Mines began with an overview of the difficulties in achieving bit-reproduciblity given the chaotic nature of climate and weather models and the many non-deterministic processes that are part of an ESM development lifecycle (e.g., porting to new machines, compiler changes, optimizations, heterogeneous computing). Hammerling argued that rather than determine whether new results from a non bit-reproducible modification are correct, a more tangible question to address is whether the new results are statistically distinguishable from the previous (control) results. Similarly, Francesco Carere of the Euro-Mediterranean Center on Climate Change also provided a thorough overview of the difficulties of bit-reproducibility given so many sources of non-determinism, most of which can be explained by the loss of the associativity in floating-point arithmetic (e.g., rounding error). Carere's discussion was in the context of evaluating a parallel model code used in weather and climate research (SHYFEM [42]) against its serial counterpart. Carere pointed out that while a bit-reproducibility constraint may be useful in early or selective stages of development, the constraint inhibits most all optimization techniques needed to improve computational performance.

Hammerling's recommended statistical approach evaluates new (i.e, not bit-identical) data in the context of an ensemble of ESM runs. Ensembles are quite common tools in weather and climate predictability research as they help to characterize the natural variability or uncertainty in the model system. The underlying idea is that changes to the model that should not be climate-changing can be evaluated for "correctness" by a statistical comparison to the control ensemble. Many factors must be taken given the typically high dimensionality of the variable space (for the atmospheric component of the Community Earth System Model, CESM [13], there many be more than 200 variables, many of which many be highly correlated). Hammerling provided an overview of the statistical testing framework developed by her and colleagues referred to as "Ensemble Consistency Testing" (ECT) [15, 43]. This approach, which can be thought of as hypothesis testing combined with Principal Component Analysis (PCA), has been successfully implemented and used for the last few years. Teo Price-Broncucia gave a follow-up presentation describing a general methodology for analyzing the output of an arbitrary ESM in order to properly adapt the ECT framework. The effectiveness of this methodology was demonstrated using the atmospheric component of the Model Across Prediction Scales (MPAS-A) [49].

Another ensemble-based method was presented by Salil Mahajin of Oak Ridge National Laboratory. His motivation was also based on the fact that near-exascale machines require code transformations to Earth System Models that may not reproduce the original model solution bit-for-bit and the resulting need to differentiate round-off error from systematic bugs. Mahajin's ensemble approach for DOE's Energy Exascale Earth System Model (E3SM) [38] consists of a series of non-bit-for-bit tests that use two-sample equality of distribution tests (e.g., K-S tests) to compare ensembles created the both the new and control codes. These tests evaluate statistical reproducibility of atmosphere and ocean (see, e.g., [40, 41]. Michael Kelleher, also from Oak Ridge, gave a follow-up presentation that addressed reducing false detections from the two-sample tests in E3SM nightly testing using false discovery rate corrections.

The statistical approach advocated in Carere's presentation for a hydrodynamic model is also an ensemble-based approach that uses the output from an ensemble of simulations to estimate rounding error. In this approach, confidence intervals of the distributions obtained by ensemble experiments must overlap to establish the parallel reproducibility of a serial run. While a number of different statistical approaches were discussed, it is clear that all share the belief that relying on bit-identical output to encore correctness is no longer practical in today's landscape of rapidly developing codes and heterogeneous architectures.

5.3 Testing

Software testing is vital for ensuring correctness and reproducibility in climate and weather modeling, making it one of the most utilized techniques in the community. The workshop covered various testing approaches, including unit testing, integration testing, and regression testing, along with some unique and newly emerging methods.

Unit testing: This technique targets specific portions of code, such as functions or subroutines, to ensure their correctness. While effective, it remains less prevalent in scientific computing applications. Edward Hartnett, one of the workshop presenters advocating for unit testing, showcased the methodology for the NCEPLIBS library [10], highlighting benefits like mitigating software failure risks and enhancing code maintainability and extensibility.

Property-based testing: Participants discussed property-based testing as a complementary approach to unit testing. Although less explored in the climate and weather modeling community, property-based testing operates similarly to unit testing but automatically explores a broader input space. It offers increased coverage compared to unit testing, and may be useful especially when a large number of inputs may be needed to discover corner cases.

Integration and System tests: Unlike unit testing, integration tests and system tests assess the interaction among various model components and the overall functionality of the software system. Due to the component-based structure inherent in climate and weather systems, these tests are indispensable and extensively employed in both operational and research-oriented climate and weather codebases. It is worth noting, however, that system-wide tests are significantly more computationally demanding and are more challenging to interpret when issues are caught.

Component-level tests: During the workshop, component-level testing was introduced by Thomas Clune, a GEOS [27] developer, as a middle-ground between unit tests and system tests to alleviate the limitations of unit testing, such as challenges in achieving comprehensive coverage and maintenance, as well as the constraints of system testing, particularly in isolating defects with specificity. This approach, which considers the hierarchical structure of GEOS components, requires less computational resources compared to system testing and provides more localized coverage. However, a challenge lies in acquiring inputs and expected outputs for each component to be tested.

Regression tests: These tests involve executing a test suite both before and after implementing code changes, comparing the results to detect any unexpected alterations in behavior. Widely relied upon in climate and weather software, their applicability may be constrained by the chaotic nature of these models and the diverse array of software and hardware systems they operate on.

Custom tests: Participants also discussed unique testing methodologies tailored to their applications. Marshall Ward from the MOM6 [9] consortium touched on two such tests routinely carried out by the MOM6 developers. For instance, the dimensional consistency test relies on uniformly scaling arithmetic equation operands to arbitrary powers of two, depending on the units of operands, and verifying the model yields identical results with and without the application of these scales. Another novel test used by the MOM6 developers, called the rotational consistency test, involves rotating the computational domain and all fields to ensure consistent results.

In addition to these various testing methodologies, participants stressed the efficacy of test-driven development, wherein tests are written prior to or during model development in a proactive manner.

5.4 Development Practices

Workshop participants have shared various model development practices and strategies aimed at ensuring both correctness and reproducibility. For instance, the MOM6 consortium emphasizes the importance of maintaining bit reproducibility by constraining model development to unambiguous floating-point operations. This includes the explicit use of parentheses in calculations involving three or more operands to address the non-associativity issue. Furthermore, efforts are made to reduce reliance on transcendental functions dependent on external libraries, which may lack bitwise consistency across versions. Emphasizing the reproduction of sum operations is also underscored [31].

Additionally, MOM6 developers stress the significance of well established version control practices such as descriptive commit logs, focusing on explaining the rationale behind changes rather than just listing modifications, to provide a detailed code evolution history and simplify debugging. The practice of routinely rebasing commit history is also endorsed to establish a more manageable linear history, aiding in bug tracking. The MOM6 consortium's structure employs a hub-and-spoke repository model with several main partners, guiding users to engage with partners instead of directly with the main fork, promoting an agile and structured development environment.

Similarly, the developers of the METplus [8] verification tool strive for software correctness through comprehensive code review processes involving multiple reviewers including scientists, engineers, and documentation specialists. Development progress is tracked using GitHub issue assignments, while continuous integration is ensured via GitHub Actions. Additionally, the project maintains clearly defined

release cycles and versioning to enhance reproducibility, which is further supported by documentation and consistent sample data.

As for the UFS weather model [12], challenges towards correctness were noted as a high level of system complexity, significant code updates (including legacy software), shared dependencies, and the absence of unit testing, among others. Proposed and implemented solutions include categorizing code changes based on their invasiveness and adopting a hierarchical testing framework encompassing unit testing, integration testing, system testing, and acceptance testing. One interesting future direction that was mentioned involves exploring machine learning applications for verifying code updates that may impact results.

Finally, a general theme across many presentations were the adoption of continuous integration and continuous deployment (CI/CD) practices, such as those utilized on GitHub, to streamline development, testing, and release cycles.

5.5 Analysis tools and benchmarks

Verification packages can play a crucial role in assessing model correctness in a portable manner. One such example tool that was presented in the workshop is METplus [8], an interagency initiative for unified model verification. METplus serves as a versatile verification framework covering a broad spectrum of temporal and spatial scales. Its design allows for extensibility through additional capabilities developed by the community. By offering a comprehensive and unified verification tool accessible to various user groups such as forecasters, university researchers, and national labs, METplus facilitates a smoother transition from research to operational applications through a consistent set of metrics. This unified approach encourages the adoption of a common verification language and provides an opportunity to train users in verification best practices.

Similarly, diagnostic tools are vital for maintaining the scientific and software integrity of climate and weather applications. One such tool discussed in the workshop is ESMValTool [3], a collaborative effort designed for analyzing data generated by Earth System Models (ESMs). The developers emphasize modular design as a critical strategy for ensuring tool reliability. This includes separating frequently used functionalities from one-time tasks and implementing rigorous unit, integration, and regression testing. Additionally, they leverage existing, well-maintained libraries whenever possible and employ custom comparison techniques for NetCDF files to reduce the need for manual testing.

Another similar means of fostering collaborative verification efforts is the utilization of benchmarks. Benchmarks are already utilized in the broader HPC community to assess performance and evaluate correctness issues stemming from parallelism e.g., message passing, shared memory parallelization, etc [45, 37]. These can also be utilized for climate and weather modeling-specific verification efforts. The High Performance Climate and Weather (HPCW) benchmark [5] is one such example that was presented in the workshop. This benchmark comprises a collection of realistic, near-operational weather and climate workloads and encompasses popular models like ICON [35], IFS [6], and NEMO [11]. Its workflow incorporates test cases, verification procedures, and scoring metrics for comparative analysis, all within a model-agnostic and customizable framework. This framework, based on the principle of separation of concerns, is equipped with flexible tools such as the SPACK package manager.

5.6 Domain-Specific Languages

Within the realm of numerical simulation software, Domain-Specific Languages (DSLs) are being increasingly explored as tools to enhance the separation of concerns and thereby streamline correctness and reproducibility efforts [22, 30]. DSLs enable the separation of high-level specifications from the lower-level software infrastructure tasks such as parallelization, discretization, and input/output operations. One such DSL for weather and climate models, called GT4Py [22, 4], was presented by climate modelers working on ICON [35], a weather prediction and climate modeling system. The developers detailed their

use of GT4Py and how it facilitated the adoption of a tiered approach to verification and validation efforts. This approach allows for the allocation of distinct expertise across various layers of the application, with software engineers focusing on lower-level software details and domain scientists concentrating solely on validation and analysis of the modeled climate. Their tiered verification approach includes runtime verification and statistical testing on grid-cell level ensemble simulations, demonstrating its effectiveness in identifying and correcting errors introduced during development.

5.7 Formal Methods and Verification Tools

Formal methods, which are mathematically rigorous techniques for specifying and verifying software systems, were another point of discussion at the workshop. These methods have proven highly effective in identifying flaws, particularly in safety-critical systems within the industry [19, 36, 44]. Despite being perceived as less accessible compared to other approaches, formal methods hold significant potential for verifying scientific software. This is because they rely on specification of abstract representations of software systems, aligning well with the accustomed abstract thinking and modeling processes of scientists and engineers [18]. Moreover, their basis in rigorous and exhaustive analysis enables them to identify flaws that may elude even the most comprehensive test suites. John Baugh's keynote underscored this perspective, emphasizing the management of complexity, e.g., the intricacies of floating point operations, through abstraction. It presented contrasting opinions on the utilization of formal methods, debating between integrating them seamlessly into familiar environments like compilers [48] versus advocating for their direct use by developers from the software design phase onwards, emphasizing a focused and streamlined approach known as lightweight formal methods [33]. The discussion highlighted the "small scope hypothesis" by Daniel Jackson, suggesting that a significant portion of bugs can be detected by testing all inputs within a limited scope [34]. Additionally, it underscored the importance of considering the interstitial machinery within scientific programs, often overlooked in reasoning about model correctness alongside numerical expressions. The talk showcased tools like Alloy [1] for automatic exploration of complex system formations, such as ocean circulation models and pipe network systems, aiding in identifying flaws in conceptual representations of physical systems in scientific software.

At a similarly themed talk by Dominic Orchard, the focus was on the role of verification and static analysis in scientific software development, as well as an overview of contemporary techniques, particularly relevant to the climate and weather modeling community. The discussion delved into the limitations of testing while expressing optimism about emerging approaches like property-based testing for comprehensive exploration of input spaces. Moreover, the talk explored the efficacy of various techniques including type systems, deductive verification, static analysis tools, proof assistants, interactive theorem provers, and model checking for thorough verification processes. The speaker acknowledged the challenges of full formal verification for climate models but highlighted lightweight formal methods and static analysis as more feasible alternatives. Tools like CamFort [2] were cited for their ability to offer verification based on static analysis, focusing on properties like numerical stability and dimensional consistency. Emphasizing the promise of static analysis as a future direction, the talk underscored its accessibility and targeted analysis capabilities. The discussion contextualized these advancements within the broader framework of program verification in computer science, pointing out the gap between theoretical advancements and their practical deployment in scientific communities such as climate science.

5.8 ML/AI Talks

Ganesh Gopalakrishnan and Harvey Dam, both of University of Utah, gave a joint talk addressing challenges in correctness for both HPC and ML. They opened with a nice overview of the HPC correctness stack (e.g., see [28]) that illustrates the path from an informal then more rigorous problem description, to the mathematical formulation, followed by numerical libraries and algorithmic approximations, and finally numerical/parallel behavior. This last step involving (sometimes extreme) heterogeneous hardware

causes the most headaches for HPC correctness. CPU and GPU features are rapidly changing and reproducibility properties are often unknown, and floating-point NaN and INF exceptions are particularly problematic, especially on GPUs (e.g., see [39]). The second half of the talk addressed the question of how to determine correctness in the ML space and how does it impact science. In the ML space, correctness is very nuanced and is not just about loss functions and accuracy, but also robustness, fairness, and model complexity, and the different ways to measure these attributes.

A second talk by David John Gagne of NCAR further elaborated on the challenges of reproducibility and correctness in the context of using ML for weather models. For example, making sure workflows are consistent and evaluating the correctness of our models is especially important as we move toward incorporating ML tools into operational weather codes. A particular challenge lies in reproducing model training and predictions as well as achieving consistent performance in an operational setting. Another challenge is that rescaling data is common to most ML approaches and is typically a lossy operation in floating-point arithmetic that is not reproducible across machines. And finally, addressing how to evaluate simulations when there is high variance between different models is important.

Similar to Gagne's presentation, Peter Dueben of ECMWF also reflected on the difficulty of achieving correctness and reproducibility when using ML approaches in weather and climate. In particular, it is not only the ML models which should be published and openly accessible, but also, and perhaps more importantly, the datasets used to train these models. Oftentimes, the typical application scenario of ML models is that they get trained on different, merged or manipulated input datasets, and, therefore, the provenance of the training dataset becomes hard to comprehend. A solution to this problem could be to establish the creation and use of benchmark datasets for specific aspects of the physical system represented by the ML models. These benchmark datasets would be openly available, supplied with reference ML solutions, evaluation metrics and computational benchmarks [23]. Currently, such an approach is developed and tested by the scientific community in the context of the Maelstrom Project [7]. Taking a step back and reflecting on the overall advance of ML approaches in all areas of science, Dueben made the point that no matter how good the ML models appear, they will always have to be evaluated by and developed with experienced members of the scientific community in order to achieve best results. Since the largest and most promising ML applications in the field of weather and climate are currently developed and maintained by big technology companies (e.g. Google, Huawei, NVIDIA), often in close collaboration with domain scientists, this approach might require a shift in the overall working philosophy of scientists in terms of public/private partnerships.

6 Panel: Correctness and Verification across platforms

The final session of the workshop was an engaging panel on correctness and verification across platforms. The four panelists and moderator are listed in Table 2. Questions were asked both by the moderator and the audience. Here we will briefly summarize several of the main discussion points addressed by the panel.

The first question offered panelists the opportunity to share their thoughts on a common practice in computational science for climate and weather modeling that they felt was problematic and/or needed addressing. The following are some of the topics raised and discussed:

- Unit tests and property tests can be very powerful tools for developers, and are not as commonly used as they ought to be.
- The issue of defining whether a model is correct in the development process is nontrivial, particularly when there is not data available to compare with (like new ultra-high resolution runs, for example).
- FAIR (Findability, Accessibility, Interoperability, and Reusability) data principles [51] and reproducible workflows were rarely mentioned in this workshop, yet are critically important for this community.

- Data races are going to cause non-determinism and non-reproducibilty, and we don't hear as much about this as the effects of floating-point differences. Furthermore, no tools are currently even available for detecting data races on GPUs.

- The manner in which compilers are developed and supported by different companies and groups hinders science. In particular, competing business objectives hinder code interoperability, and this obstacle can prevent scientists from writing code that is both portable and performant across architectures.

The second topic raised by the moderator concerned the massive diversification of computing environment, which includes everything from a variety of GPUs, to heterogeneous nodes, to AI/ML hardware, novel floating point formats, and even lossy-compressed data. The question was how this diversification changes approaches to reproducibility, correctness and verification for climate and weather models and what needs to be considered for best practices moving forward. The following are some of the topics raised and discussed:

- We are developing models with different infrastructure and using different compilers and different hardware. The pace of development is often rapid, and too often there is just not a good structure for the entire development workflow.

- Mixed-precision tuning is increasingly popular, particularly with the popularity of GPUs. We must be careful with it as it can cause correctness issue. It's important to note that contrary to popular opinion, ML code is not immune to low-precision issues, such as NaNs (e.g., see[32]).

- The end of Moore's law is driving the increase in diversity of hardware, and chips have become more advanced and compact. As a result, silent data corruption will be more common as hardware becomes more dense, and this issue has not received enough attention. We too often assume that our machine is giving the right answer, but a hardware defect or issue may result in the wrong answer. Such errors are very difficult to detect.

- In many cases, the relationship between software engineers and climate model developers should be stronger. In some sense, a more heterogeneous project team is very helpful for figuring out the cause of heterogeneous issues.

- The community really needs to commit to deciding on compression guidelines and advice to address the storage issues arising from faster computation and higher resolution models.

- Another important issue that deserves more attention in the community is data integrity. How can we be sure that our data has not been tampered with (e.g., by AI or something)? For example, CESM has checks on input data for simulation runs to make sure the data has not been altered, but what about output data sets?

Another interesting discussion centered on how much reproducibility we currently have or need for climate and weather models. We list a few of the salient points that were raised:

- Reproducing the entire software/hardware stack is not practical after a certain amount of time has passed. If the machine that doesn't exist anymore and the compiler that doesn't exist anymore, then expecting exact reproducibility is unrealistic.

- The reproducibilty that we want and need depends a lot on timescale. Do we want the same answer in 10 years for a climate simulations? Probably not, as we would expect scientific progress to have been made.

- Reproducibility is not always possible even now with certain applications and hardware. Often the algorithm designers have to decide what is possible.

- In terms of papers, sometimes you can't save everything that is needed to reproduce every step in the workflow.
- Containers cannot solve everything, but can be helpful in some cases.
- We have to be careful not to mix the two meanings of reproducibility: (1) bit-for-bit (BFB) reproducibility and (2) scientific reproducibility, meaning that other scientists should be able to reproduce the results. The former is quite helpful for finding bugs during development, but has limited scope in practice. The latter is something that we certainly don't need to be able to do in 10 years, and its questionable whether anyone really does this (try to reproduce someone else's results with their code/data).
- The value of publishing your data and code with a paper is important in terms of improving author diligence (e.g., in case other people *might* run your code). In practice, not many folks actually do this, and are instead more likely to write their own code to try to obtain similar results.
- An entirely different meaning of reproducibility is performance reproducibility. How important is this to the community?
- We should also keep in mind the question of whether the diagnostic tools are correct. (Maybe the model is correct, but the diagnostics tools used to look at the model data are not.) There are potentially many problems in the diagnostic space in terms of reproducibility,

Another major topic of discussion concerned funding. In particular, panelists were asked, given how complex this space is getting, how one should go about convincing management or funding agencies about the importance of adequately resourcing work in this area. This question is especially relevant for legacy codes, and several topics were discussed:

- An issue is that funding agencies, like the NSF, are highly focused on advancing the science. (And tend not to be concerned about random bit flips or race conditions on GPUs). Lack of funding for software engineering and code maintenance in general, as well as particular issues like correctness and reproducibility, is a serious problem.
- Perhaps what is needed is an honest conversation about the real cost of chasing bugs (e.g., due to lack of resources for unit testing) or the cost of determining what has gone wrong with a code on a new machine (because proper reproducibilty measures were not put in place).
- Data management plans are now mandatory in many situations. For example, they are often a part of the funding process. Could we similarly require a plan for the correctness and reproducibility of codes?
- One challenge is the changing composition of the HPC community. In particular, vendors and the HPC community used to work very closely work together. But now many companies with the deepest pockets are not doing hardware development for the HPC community anymore as they pivot toward targeting the AI space and customers.

A final topic from the panel that we mention here is how do domain scientists and graduate students fit into the big picture. So many issues have been brought up at this workshop, and one person cannot begin to be familiar with everything that there is to worry about. The following ideas around this topic were discussed:

- A large code needs an automated system for new code developers as well as developer test suites. In other words, tools are needed to empower new developers to feel confident about their code.

- Containers can certainly be helpful in this context, allowing a graduate student, for example, to more easily run a model without worrying about porting it to their own hardware/software stack.
- We need to increase awareness in graduate courses, for example. Curriculum may have to change to adapt as scientists need to know more about software development.
- Related to the previous discussion, it is hard to secure funding toward efforts to increase ease-of-use and usability. This situation is unfortunate as investments for ease-of-use will increase productivity for everyone.
- Some model codes are more accessible than others. For example, DOE is not focused on the community being able to run their climate model, whereas CESM places more value in being accessible to the community.

7 Concluding Remarks

The Workshop on Correctness and Reproducibility for Climate and Weather Software was held on November 9-10, 2023 at the National Center for Atmospheric Research in Boulder, CO. A diverse group of participants from NCAR, universities, research labs, and industry met, including researchers from both the US and international institutions. In particular, application scientists and model developers together with computer scientists with expertise on compilers, correctness and reproducibility, shared experiences, best practices, and challenges.

This workshop raised awareness of correctness and reproducibility issues for model simulations across NCAR labs and programs. Furthermore, climate-and-weather-focused needs in these areas received increased visibility in the broader research community.

7.1 Key takeaways

We begin by summarizing several key challenges identified, which complicate ensuring correctness and reproducibility efforts in climate and weather modeling:

- **Software Complexity:** Models are sophisticated software systems resulting from decades of development. They incorporate a variety of complex components, often including legacy code.
- **Computational Intensity:** Running these models demands significant computing resources. The increasingly heterogeneous HPC environments and the shift towards higher resolution configurations further exacerbate the computational cost.
- **Defining "Correctness" and "Reproducibility":** With the well known aphorism "all models are wrong but some are useful" in mind, the climate and weather modeling community needs clearer definitions of "correctness" and "reproducibility" along with classifications outlining different levels of these concepts.
- **Development Practices:** The lack of robust development practices within the climate and weather modeling community hinders efforts to achieve correctness and reproducibility.
- **Funding Priorities:** Securing funding for software verification and maintenance is challenging, despite their crucial role in model accuracy and reliability.
- **Machine Learning:** The increasing use of machine learning (ML) in climate modeling and weather prediction introduces new complexities in evaluating correctness and ensuring reproducibility. Correctness and reproducibility efforts must adapt to accommodate these ML-based approaches.

- **Compiler Development and Support:** The development and support of compilers are often driven by competing business objectives hindering code interoperability. This obstacle prevents scientists from writing code that is both portable and performant across different architectures.
- **Silent Data Corruption:** With the increase in scale and diversity of hardware, silent data corruption will become more common. This issue has not received enough attention. We often assume our machines provide correct answers, but hardware defects or issues may produce errors that are very difficult to detect.

In light of these challenges, several key opinions, solutions, and future directions were discussed during the workshop:

- **Scope of Applicability:** Correctness should be defined by the "scope of applicability" for a model, identifying the specific aspects of the physical system it aims to represent. This approach allows for coherent explanations and predictions within the model's scope while acknowledging the errors inherent in numerical modeling.
- **Software Engineering Tools:** Improving software correctness involves adapting software engineering tools and processes to make scientists more effective at writing software.
- **Interdisciplinary Collaboration:** Stronger integration between software engineers and domain scientists is essential. A heterogeneous project team can better address diverse issues.
- **Bit-Reproducibility:** While useful in early development, relying on bit-identical output for correctness is impractical with rapidly developing codes and heterogeneous architectures.
- **Funding Focus:** Funding agencies tend to prioritize advancing science over addressing maintenance and code quality issues like random bit flips or race conditions on GPUs. The lack of funding for software engineering and code maintenance is a serious problem. It is therefore essential to communicate the importance of robust software practices to funding agencies.
- **Educational Awareness:** Modeling centers and graduate programs need to increase awareness and adapt curricula to teach the next generation of model developers more about software development and best practices.
- **Automated Systems for New Developers:** Large codebases need automated systems and developer test suites to empower new developers to feel confident about their code.
- **Ensemble-Based Statistical Consistency Testing:** This technique has proven highly effective for assessing correctness by confirming that the overall statistical properties of model outputs are consistent, especially when bitwise reproducibility cannot be achieved.
- **Software Testing:** Vital for ensuring correctness and reproducibility, software testing is one of the most utilized techniques in the climate and weather modeling community. While integration and system tests are common, unit testing is less utilized but can significantly reduce software failure risks and enhance code maintainability and extensibility. Test-driven development, where tests are written prior to or during model development, is also highly effective.
- **Version Control and CI/CD Practices:** Well-established version control practices, regular release cycles, comprehensive code review processes, thorough documentation, and the adoption of continuous integration and continuous deployment (CI/CD) practices, such as those utilized on GitHub, streamline development, testing, and release cycles, thereby streamlining correctness and reproducibility efforts.

- **Rigorous Techniques:** Lightweight formal methods and static analysis offer significant potential for verifying scientific software wherever feasible. Their basis in rigorous and exhaustive analysis enables the identification of flaws that may elude even the most comprehensive test suites.
- **Standardized Diagnostic and Analysis Tools:** By providing standardized and portable diagnostics and analysis tools, comprehensive and comparative correctness and reproducibility efforts can be facilitated across modeling groups.

7.2 Participant Feedback

The feedback received on the workshop was resoundingly positive. A final survey was emailed to all attendees after the workshop, and 25 participants responded. For the question on rating the overall quality of the workshop, 76% of the respondents gave a rating of 5, and the remainder gave a score of 4. (The scale was 1 to 5, with 1 being "poor" and 5 being "excellent"). Similarly encouraging was that 76% answered that they were very likely to attend a future event. (Again, the scale was 1 to 5, with 1 being "unlikely" and 5 being "very likely".)

The survey responses included a number of compliments about the workshop content, the speakers, the panel discussion, and overall organization. Furthermore, a number of suggestions for a future event were given, including the following:

- Perhaps a bit more time per speaker (maybe the entire workshop should be more than 2 days in length).
- More focus on the "why" from modelling centers (not just the "what").
- More structured social time for interaction (maybe a banquet in the evening).
- More on uncertainty quantification and computational reproducibility.
- Invite more people from outside of the domain (e.g., computer science).
- It would be nice to see more graduate student participants.
- More machine learning talks.
- Prefer that speakers are all in person.
- Suggest a targeted discussion on best practices for achieving reproducibility in the weather/climate model field.
- Proximity (time and location) to SC23 may not have been as important as we anticipated (only 5 of 25 survey respondents attended SC23 in Denver), though it did benefit international attendees.

7.3 Next Steps

Overall, the attendees with whom we interacted and the survey results indicate that this workshop was a strong success on several fronts. The quality of speakers, the variety of talks, and the general organization all received positive feedback. Further, this workshop provided beneficial opportunities for community members to interact, supporting NCAR's community-focused mission.

We, the organizers, are interested in hosting a follow-up workshop in future and have already received inquires about our plans. Our current thoughts are to host the workshop every other year, targeting the fall of 2025 for the next one. In the meantime, we will look for a new source of funding for the workshop. Hosting the workshop at NCAR again would allow us to keep costs low, though SC25 will be in St. Louis. A modest amount of funding is needed to fund travel for keynote speakers and panelists to

promote in-person participation. We also need travel funding to ensure graduate student participation. Finally, snacks and beverages are important for the productive networking breaks. Some costs could be covered by registration fees, though we prefer to keep those minimal.

Acknowledgments

We thank all of the NCAR staff and community members who attended this workshop (both in-person and virtually) and contributed their ideas and perspectives to this effort. Funding for the workshop was provided by the UCAR President's Strategic Initiative Fund (PSIF) and co-sponsorship from both the Computational and Information Systems Lab and the Climate and Global Dynamics Lab at NCAR. We would also like to thank the invaluable administrative support provided for the workshop by Taysia Peterson, Kristen Pierri, Lisa Larson, and Mary Pronk.

Appendices

A Workshop Program

The Workshop on Correctness and Reproducibility for Climate and Weather Software consisted of four keynote talks, 17 contributed talks, and a panel discusssion. The program schedule, talk slides, and recordings are available on the workshop website: `https://ncar.github.io/correctness-workshop/`.

Table 1: Keynote speakers and talk titles.

Keynote Talks	
Steve Easterbrook School of the Environment and Department of Computer Science, University of Toronto	*Models, Data, and Wisdom: How do we know when to trust a climate model?*
Peter Dueben Earth System Modelling Section, European Centre for Medium Range Weather Forecasts	*Earth system models of the future*
John Baugh Civil Engineering and Operations Research, North Carolina State University	*Lightweight Formal Methods: The What, Why, and How*
Dorit Hammerling Applied Mathematics and Statistics, Colorado School of Mines	*Contained Chaos: Quality Assurance for the Community Earth System Model*

Table 2: Panel participants.

Panel: Correctness and verification across platforms	
Panelists:	
Ilene Carpenter	Hewlett Packard Enterprise
Karsten Peters-von Gehlen	Deutsches Klimarechenzentrum GmbH (DKRZ)
Ganesh Gopalakrishnan	University of Utah
Aaron Donahue	Livermore National Laboratory
Moderator:	
Brian Dobbins	National Center for Atmospheric Research

References

[1] Alloy. https://alloytools.org/. Accessed: 2012-06-1.

[2] Camfort. https://camfort.github.io/. Accessed: 2012-06-1.

[3] ESMValTool. https://esmvaltool.org/. Accessed: 2012-06-1.

[4] GT4Py: GridTools for Python . https://github.com/GridTools/gt4py. Accessed: 2012-06-1.

[5] HPCW - The High Performance Climate Weather Benchmark. https://www.esiwace.eu/services/hpcw. Accessed: 2012-06-1.

[6] IFS: Integrated Forecasting System. https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model. Accessed: 2012-06-1.

[7] The Maelstrom Project. https://www.maelstrom-eurohpc.eu/. Accessed: 2012-06-1.

[8] METplus. https://github.com/dtcenter/METplus. Accessed: 2012-06-1.

[9] MOM6. https://noaa-gfdl.github.io/MOM6/. Accessed: 2012-06-1.

[10] NCEPLIBS. https://github.com/NOAA-EMC/NCEPLIBS. Accessed: 2012-06-1.

[11] NEMO: Nucleus for European Modelling of the Ocean. https://www.nemo-ocean.eu/. Accessed: 2012-06-1.

[12] UFS Weather Model. https://epic.noaa.gov/get-code/ufs-weather-model/. Accessed: 2012-06-1.

[13] The Community Earth System Model Version 2 (CESM2). Journal of Advances in Modeling Earth Systems, 12(2):e2019MS001916, 2020. ISSN 1942-2466. doi: 10.1029/2019MS001916. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001916.

[14] D. Ahn, A. Baker, Michael Bentley, Ian Briggs, G. Gopalakrishnan, D. Hammerling, I. Laguna, G. Lee, D. Milroy, and M. Vertenstein. Keeping science on keel when software moves. Communications of the ACM, 64:66 – 74, 2021.

[15] A. H. Baker, D. M. Hammerling, M. N Levy, H. Xu, J. M. Dennis, B. E. Eaton, J. Edwards, C. Hannay, S. A. Mickelson, R. B. Neale, D. Nychka, J. Shollenberger, J. Tribbia, M. Vertenstein, and D. Williamson. A new ensemble-based consistency test for the Community Earth System Model. Geoscientific Model Development, 8:2829–2840, 2015. doi: 10.5194/gmd-8-2829-2015.

[16] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. Nature, 525(7567):47–55, 2015. ISSN 0028-0836. doi: 10.1038/nature14956. URL http://dx.doi.org/10.1038/nature14956.

[17] Peter Bauer, Peter Dominik Dueben, Torsten Hoefler, Tiago Quintino, Thomas C. Schulthess, and Nils P. Wedi. The digital revolution of earth-system science. Nature Computational Science, 1:104 – 113, 2021. URL https://api.semanticscholar.org/CorpusID:234024066.

[18] John Baugh and Alper Altuntas. Formal methods and finite element analysis of hurricane storm surge: A case study in software verification. Science of Computer Programming, 158:100–121, 2018.

[19] Nikolaj Bjørner and Karthick Jayaraman. Checking cloud contracts in Microsoft Azure. In Distributed Computing and Internet Technology: 11th International Conference, ICDCIT 2015, Bhubaneswar, India, February 5-8, 2015. Proceedings 11, pages 21–32. Springer, 2015.

[20] Zied Ben Bouallègue, Mariana C A Clare, Linus Magnusson, Estibaliz Gascón, Michael Maier-Gerber, Martin Janoušek, Mark Rodwell, Florian Pinault, Jesper S Dramsch, Simon T K Lang, Baudouin Raoult, Florence Rabier, Matthieu Chevallier, Irina Sandu, Peter Dueben, Matthew Chantry, and Florian Pappenberger. The rise of data-driven weather forecasting: A first statistical assessment of machine learning-based weather forecasts in an operational-like context. Bulletin of the American Meteorological Society, 2024. doi: 10.1175/BAMS-D-23-0162.1. URL https://journals.ametsoc.org/view/journals/bams/aop/BAMS-D-23-0162.1/BAMS-D-23-0162.1.xml.

[21] T. Clune and R. Rood. Software testing and verification in climate model development. IEEE Software, 28(6):49–55, 2011. doi: http://dx.doi.org/10.1109/MS.2011.117.

[22] Johann Dahm, Eddie Davis, Tobias Wicky, Mark Cheeseman, Oliver Elbert, Rhea George, Jeremy James McGibbon, Linus Groner, Enrique Paredes, and Oliver Fuhrer. Gt4py: Python tool for implementing finite-difference computations for weather and climate. In 101st American Meteorological Society Annual Meeting. AMS, 2021.

[23] Peter D. Dueben, Martin G. Schultz, Matthew Chantry, David John Gagne, David Matthew Hall, and Amy McGovern. Challenges and benchmark datasets for machine learning in the atmospheric sciences: Definition, status, and outlook. Artificial Intelligence for the Earth Systems, 1(3):e210002, 2022. doi: 10.1175/AIES-D-21-0002.1.

[24] Steve M. Easterbrook. Computing the Climate: How We Know What We Know About Climate Change. Cambridge University Press, 2023.

[25] Steve M. Easterbrook and Timothy C. Johns. Engineering the software for understanding climate change. Computing in Science and Engineering, 11(6):65–74, 2009. ISSN 0740-7475. doi: http://doi.ieeecomputersociety.org/10.1109/MCSE.2009.193.

[26] Steve M. Easterbrook, Paul N. Edwards, Venkatramani Balaji, and Reinhard Budich. Guest editors' introduction: Climate change - science and software. IEEE Software, 28(6):32–35, 2011. doi: 10.1109/MS.2011.141.

[27] GEOS contributors. GEOS computational geometry library. Open Source Geospatial Foundation, 2021. URL https://libgeos.org/.

[28] Maya Gokhale, Ganesh Gopalakrishnan, Jackson Mayo, Santosh Nagarakatte, Cindy Rubio-González, and Stephen F. Siegel. Report of the DOE/NSF Workshop on Correctness in Scientific Computing, June 2023, Orlando, FL, 2023.

[29] Ganesh Gopalakrishnan, Paul D. Hovland, Costin Iancu, Sriram Krishnamoorthy, Ignacio Laguna, Richard A. Lethin, Koushik Sen, Stephen F. Siegel, and Armando Solar-Lezama. Report of the HPC correctness summit, Jan 25-26, 2017, Washington, DC. CoRR, abs/1705.07478, 2017. URL http://arxiv.org/abs/1705.07478.

[30] Tobias Gysi, Carlos Osuna, Oliver Fuhrer, Mauro Bianco, and Thomas C Schulthess. Stella: A domain-specific tool for structured grid methods in weather and climate models. In Proceedings of the international conference for high performance computing, networking, storage and analysis, pages 1–12, 2015.

[31] Robert Hallberg and Alistair Adcroft. An order-invariant real-to-integer conversion sum. Parallel Computing, 40(5-6):140–143, 2014.

[32] Yi He, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, and Yanjing Li. Understanding and mitigating hardware failures in deep learning training systems. In Proceedings

of the 50th Annual International Symposium on Computer Architecture, ISCA '23. Association for Computing Machinery, 2023. doi: 10.1145/3579371.3589105. URL https://doi.org/10.1145/3579371.3589105.

[33] Daniel Jackson. Lightweight formal methods. In FME 2001: Formal Methods for Increasing Software Productivity: International Symposium of Formal Methods Europe Berlin, Germany, March 12–16, 2001 Proceedings, pages 1–1. Springer, 2001.

[34] Daniel Jackson. Alloy: a language and tool for exploring software designs. Communications of the ACM, 62(9):66–76, 2019.

[35] Johann H Jungclaus, Stephan J Lorenz, Hauke Schmidt, Victor Brovkin, Nils Brüggemann, Fatemeh Chegini, Traute Crüger, Philipp De-Vrese, Veronika Gayler, Marco A Giorgetta, et al. The ICON earth system model version 1.0. Journal of Advances in Modeling Earth Systems, 14(4): e2021MS002813, 2022.

[36] Roope Kaivola, Rajnish Ghughal, Naren Narasimhan, Amber Telfer, Jesse Whittemore, Sudhindra Pandav, Anna Slobodová, Christopher Taylor, Vladimir Frolov, Erik Reeber, et al. Replacing testing with formal verification in intel coretm i7 processor execution engine validation. In International Conference on Computer Aided Verification, pages 414–429. Springer, 2009.

[37] Mathieu Laurent, Emmanuelle Saillard, and Martin Quinson. The MPI bugs initiative: a framework for MPI verification tools evaluation. In 2021 IEEE/ACM 5th International Workshop on Software Correctness for HPC Applications (Correctness), pages 1–9. IEEE, 2021.

[38] L. Ruby Leung, David C. Bader, Mark A. Taylor, and Renata B. McCoy. An introduction to the e3sm special collection: Goals, science drivers, development, and analysis. Journal of Advances in Modeling Earth Systems, 12(11):e2019MS001821, 2020. doi: https://doi.org/10.1029/2019MS001821.

[39] Xinyi Li, Ignacio Laguna, Bo Fang, Katarzyna Swirydowicz, Ang Li, and Ganesh Gopalakrishnan. Design and evaluation of GPU-FPX: A low-overhead tool for floating-point exception detection in NVIDIA GPUs. In Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing, HPDC '23, page 59–71, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701559. doi: 10.1145/3588195.3592991. URL https://doi.org/10.1145/3588195.3592991.

[40] Salil Mahajan, Abigail L. Gaddis, Katherine J. Evans, and Matthew R. Norman. Exploring an ensemble-based approach to atmospheric climate modeling and testing at scale. Procedia Computer Science, 108:735–744, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.05.259. URL https://www.sciencedirect.com/science/article/pii/S1877050917308906. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[41] Salil Mahajan, Katherine J. Evans, Joseph H. Kennedy, Min Xu, and Matthew R. Norman. A multivariate approach to ensure statistical reproducibility of climate model simulations. In Proceedings of the Platform for Advanced Scientific Computing Conference, PASC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367707. doi: 10.1145/3324989.3325724. URL https://doi.org/10.1145/3324989.3325724.

[42] G. Micaletto, I. Barletta, S. Mocavero, I. Federico, I. Epicoco, G. Verri, G. Coppini, P. Schiano, G. Aloisio, and N. Pinardi. Parallel implementation of the SHYFEM (System of HydrodYnamic Finite Element Modules) model. Geoscientific Model Development, 15(15):6025–6046, 2022. doi: 10.5194/gmd-15-6025-2022. URL https://gmd.copernicus.org/articles/15/6025/2022/.

[43] D. J. Milroy, A. H. Baker, D. M. Hammerling, and E. R. Jessup. Nine time steps: ultra-fast statistical consistency testing of the Community Earth System Model (pyCECT v3.0). Geoscientific Model Development, 11(2):697–711, 2018. doi: 10.5194/gmd-11-697-2018. URL https://www.geosci-model-dev.net/11/697/2018/.

[44] Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. How Amazon web services uses formal methods. Communications of the ACM, 58(4):66–73, 2015.

[45] Konstantinos Parasyris, Ignacio Laguna, Harshitha Menon, Markus Schordan, Daniel Osei-Kuffuor, Giorgis Georgakoudis, Michael O Lam, and Tristan Vanderbruggen. HPC-MixPBench: an HPC benchmark suite for mixed-precision analysis. In 2020 IEEE International Symposium on Workload Characterization (IISWC), pages 25–36. IEEE, 2020.

[46] Elizaveta Pertseva, Melinda Chang, Ulia Zaman, and Michael Coblenz. A theory of scientific programming efficacy. ICSE 2024, 2024. to appear.

[47] J. Pipitone and S. Easterbrook. Assessing climate model software quality: a defect density analysis of three models. Geoscientific Model Development, 5(4):1009–1022, 2012. doi: 10.5194/gmd-5-1009-2012.

[48] John Rushby. Disappearing formal methods. In Proceedings. Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE 2000), pages 95–96. IEEE, 2000.

[49] William C. Skamarock, Joseph B. Klemp, Michael G. Duda, Laura D. Fowler, Sang-Hun Park, and Todd D. Ringler. A Multiscale Nonhydrostatic Atmospheric Model Using Centroidal Voronoi Tesselations and C-Grid Staggering. Monthly Weather Review, 140(9):3090–3105, September 2012. ISSN 1520-0493, 0027-0644. doi: 10.1175/MWR-D-11-00215.1. URL https://journals.ametsoc.org/view/journals/mwre/140/9/mwr-d-11-00215.1.xml.

[50] Tim Storer. Bridging the chasm: A survey of software engineering practice in scientific programming. ACM Comput. Surv., 50(4), Aug. 2017. ISSN 0360-0300. doi: 10.1145/3084225.

[51] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. Scientific data, 3, 2016.

Table 3: Contributed talk titles and speakers.

Contributed Talks	
Thomas Clune NASA Goddard Space Flight Center	*Component Level Regression Testing in a Hierarchical Architecture*
David Guibert Center for Excellence in Performance Programming, Eviden	*High Performance Climate and Weather Benchmark (HPCW): a framework for reproducible benchmarks of ESM models and mini-applications*
Harvey Dam, Ganesh Gopalakrishnan Department of Computer Science, University of Utah	*Correctness Challenges in HPC and ML*
Valeriu Predoi NCAS-CMS, University of Reading	*Reliable and reproducible Earth System Model data analysis with ESMValTool.*
Balwinder Singh Atmospheric Sciences and Global Change Division, Pacific Northwest National Laboratory	*Testing approach for porting legacy 4-mode Modal Aerosol Model (MAM4) to C++/Kokkos*
Abishek Gopal Institute for Atmospheric and Climate Science, ETH Zurich	*Verification of the ICON model with the GT4Py dycore - challenges and insights*
Michael Coblenz Department of Computer Science, UC San Diego	*A Theory of Scientific Programming Efficacy.*
Marshall Ward Geophysical Fluid Dynamics Lab, NOAA	*An overview of the MOM6 development cycle.*
David John Gagne Computational and Information Systems Lab, NCAR	*Challenges in Ensuring Reproducibility for Machine Learning Weather Model Training and Deployment.*
Tara Jensen Research Applications Lab, NCAR	*METplus: The Long and Winding Road to Unified Verification*
Edward Hartnett CIRES/NOAA	*Unit Testing NCEPLIBS*
Dominic Orchard Department of Computer Science and Technology, University of Cambridge and School of Computing, University of Kent	*What could the next 30 years of software verification in climate science look like?*
Francesco Carere Euro Mediterranean Center on Climate Change Foundation (CMCC Foundation)	*Parallel reproducibility of the SHYFEM-MPI model*
Teo Price-Broncucia Department of Computer Science University of Colorado Boulder	*Methods and Tools for the Application of UF-ECT to New Climate Models.*
Jun Wang NOAA NWS/EMC	*Ensure the correctness and reproducibility in UFS Weather Model CI*
Salil Mahajin Computational Earth Sciences Group, Oakridge National Laboratory	*Towards Ensuring Statistical Climate Reproducibility of Earth System Models in the Exascale Age*
Michael Kelleher Computational Earth Sciences Group, Oakridge National Laboratory	*Improvements in Reproducibility Testing Through False Discovery Rate Correction*