Precision guidelines for validation of NGEN code translation and upgrades

Overview

Differences in model performance and outcomes often arise when codes are run in different contexts -- i.e., on different operating systems or hardware, with different compilers or compiler settings, under different drivers or wrappers, and when code components are restructured. The latter context is relevant for the NGEN effort, which is modularizing components of different models into an overarching, flexible framework. There is a need for a standard to determine when simulation differences indicate a coding or implementation error, versus a difference that can be attributed to a method choice or unavoidable cause (e.g., switching from 32-bit to 64-bit hardware, switching solvers). This document proposes precision standards to be applied when accepting a code reformulation or change. These are also useful as a real-time simulation check on energy and moisture balances. An example of such standards being applied is a recent validation of an implementation of a standalone SUMMA model within the NASA Land Information System (LIS) run-time environment.

Proposed precision tolerances

Most components being included in the CFE adopt a double precision (8-byte) standard for internal state and flux variables. For efficiency, input and output variables may be single (float, 4-byte) precision. It's possible that some entire models or components that are included in the future may also have a float precision, which is not uncommon in legacy model code (and is faster than double precision code). Advanced models may even be able to switch between float and double precision implementations. Double and float variables are typically now stored with 15- and 7-digit precision (respectively), but certain operations or uses of the variables may result in an accumulation of error after multiple timesteps, and a loss of matching ability (even when code is correctly implemented) at slightly lower precision levels. *For this reason, 12 digits is proposed as a standard for validating double values, and 5 digits is proposed as a standard for float variables* (In practice, the precision for double values could be lower if float variables are involved at any stage of the calculations). While a match at these levels does not entirely rule out rule coding error in a new model version, it would give confidence that the chances of an error are low, and that if an error exists, its impact on simulation outcomes will likely be minimal even in long simulations.

Usage in NGEN

New model or code implementations will target a code validation at the proposed precision levels for key variables and mass/energy balances (eg, between a standalone model and a modularized model or component). It is not expected that all new formulations will meet the standard (everywhere and in all test timesteps), but checking

for high levels of precision will highlight where other factors (some resolvable, some not) may be impacting the portability of code from one simulation context to another. In other words, precision tolerance usage and checking is not proposed as a show-stopper for development but rather as a diagnostic tool for raising awareness about potential issues to flag and investigate when the project scope permits.

Experience to Date: Comparison of Results of NGEN CFE codes with BMI and without BMI

In our tests of the Conceptual Functional Equivalent (CFE) versus CFE implemented via the Basic Model Interface (BMI) modularization methods, earlier comparisons checked six digits after the decimal points from the formatted outputs for catchment id-87. For these comparisons, we have seen identical values between the BMI-CFE and the original CFE code for Schaake Output Runoff, GIUH Runoff, Nash Lateral Flow, Ground Water Flow, and Total Discharge, for all 720 time steps, with 1 hour time step size. These tests are performed using varying initial conditions: soil_storage_percentage at 0.22, and 0.667; groundwater_storage_percentage at 0.5 and 0.01. We also looked at the total discharge data from standard screen output, which indicated 8-9 digits agreement between BMI-CFE and CFE calculations. In earlier tests, we have also shown similar agreement between CFE calculation results and the original T-Shirt C code calculation results.

Possible reasons for precision discrepancies between CFE and BMI-CFE:

The double precision numbers have an accuracy of 15 decimal digits, the loss of accuracy from rounding off error is expected, but precisely how many digit loss probably varies and depends on the specific code. Since BMI-CFE code has additional interface and difference in coding details, we hypothesize the memory allocation, cache usage, data sets loading into the registers may be different. These likely affect the rounding off processes and cause the least significant digits to differ. According to some literature, computer hardware architecture, compiler, even optimization may all affect numerical precision.

In our CFE and BMI-CFE validation example, one particular case, deserves attention, which is: a key input variable, the rain rate, or specifically: precip_kg_per_m2, is a float point number, this could potentially reduces the precision in comparison between the two models even though all other variables are double, in consideration of rounding-off error in machine precision. There are many excellent discussions on rounding-off errors in literature due to machine precision, we refer interested readers to literature[1-4] for details.

On the other hand, the BMI-CFE and the original CFE demonstrate remarkable stability with 8-9 decimal point accuracy even after 720 time steps (hours). What are the possible reasons? We hypothesize two possible causes:

1. The codes do not use space grids, all flow values are single valued for the catchment, i.e., there is no instability introduced by spatial matrices such as those in solving finite difference equations.
2. Mass balance: In the evolution in time, the flows are constrained by mass balance. The mass balance, like energy conservation in classical molecular dynamics, should constrain the flow variation to a certain range, thus limiting the error propagation magnification in time (possibly through error cancellation).

References and Links

1. What Every Computer Scientist Should Know About Floating-Point Arithmetic.
2. Round-off error
3. Machine epsilon - Wikipedia
4. Floating-point arithmetic - Wikipedia