US Department of Commerce
National Oceanic and Atmospheric Administration (NOAA)
National Weather Service
National Centers for Environmental Prediction (NCEP)

Office Note 484

doi:10.7289/V5/ON-NCEP-484

NOAA Environmental Modeling System Regression Test Suite: System Evaluation and Requirements for a New System

Samuel Trahan
IM Systems Group (IMSG)
NCEP Environmental Modeling Center (EMC)

December, 2016

email: Samuel.Trahan@noaa.gov

Abstract

The NOAA Environmental Modeling System (NEMS) is a collection of numerical models of various aspects of the environment, coupled together via the NUOPC coupler. This creates a complex combination of source codes wherein a change to one model could break other models in unpredictable ways. For this reason, the NEMS Regression Tests (RT) were created. They compile the NEMS in various ways, and test the models in pre-specified scenarios. This suite is automated so that a user can type a simple command to run a suite of over forty such tests. When a user makes some change to one model, the user may run the test suite to see if there were any unintended effects. The present suite works well for a limited set of cases, but is having trouble adapting to new models, coupled modeling, new platforms and other issues. For this reason, we are designing a new regression test suite with increased adaptability and additional capabilities. This document describes the old system, analyzes its strengths and weaknesses, and describes requirements for a new system.

Background

The NOAA Environmental Modeling System (NEMS) consists of a NUOPC coupler and centralized build system that connect multiple models together. The coupler sends scientific data, such as temperature fields and enthalpy fluxes, between the models. This allows the models' boundary conditions to be other models, rather than constant fields or climatology. This is intended to improve forecast skill, analysis quality and nowcast skill.

The coupled combination of models has far more code to test than a single model due to creating an amalgamation of multiple models. When a developer modifies a single model, or the coupling code at the center of NEMS, it can have snowballing effects that impact results from all models, or some subset of the possible coupled combinations.

To combat this effect, the NEMS developers created a regression test suite, hereafter referred to as *rt.sh*. This regression tests suite tests prepared test cases, using the developer's copy of the model code and coupler. It compares the model output for those test cases to prior results that are known to be valid, hereafter referred to as the *baseline*.

The rt.sh has reached the limits of its capabilities in recent months, and is no longer able to meet the needs of the NEMS development community. For this reason, a new regression test suite is under development. This document will explain the strengths and weaknesses of the rt.sh, and requirements for a new suite.

Feedback Process and Acknowledgements

The requirements located later in this document came from an interactive discussion with the NEMS community including developers and users. This was done by iteratively providing a requirements list, asking for feedback, seeking management prioritization, and updating the requirements list. This was done through a combination of Google Doc interactive tools and weekly meetings. Special thanks go to the following individuals who provided extensive feedback. In alphabetical order by last name: Barry Baker, Cecelia DeLuca, Mark Iredell, Dusan Jovic, Eugene Mirvis, Jessica Meixner, Shrinivas Moorthi, Gerhard Theurich, Patrick Tripp, Ratko Vasic, Jun Wang, and Jack Woolen.

Strengths and Weaknesses of rt.sh

Lack of Modularity Causes Lack of Flexibility

The underlying implementation of rt.sh is two scripts: one for the Global Spectral Model (GSM) one for the Non-hydrostatic Multi-scale Model (NMM). Each underlying script re-implements fundamental functionality such as batch system interaction and baseline verification. Hence, rt.sh is unable to test any other models (such as standalone Wavewatch3), nor the various coupled combinations (such as GSM coupled to ocean and ice models). This could be fixed by extending the rt.sh to handle such models. However, that would require re-implementing large swaths of functionality for each model. A modular design would fix this problem; functions like baseline comparison and batch system interaction do not need to be re-written for each model.

Simplicity of User Interaction

Users of rt.sh that only want to verify their code against a baseline only need to do a few simple tasks. Execution of the rt.sh is as simple as a single UNIX shell command, "./rt.sh -s" which runs the "standard tests" (-s). One can disable or enable a test by commenting out lines of the "rt.conf" configuration file, which has an easily understood tabular syntax. The suite is divided into a "standard" test suite and a "full" test suite, each of which contain some tests for NMM and some for GSM. One can easily switch a test between standard and full. This simplicity of use is a key benefit of the current system and must be retained in any new regression test suite.

Workflow Components

The rt.sh test suite includes more than just tests of the NEMS executable. It contains components of the NCEP production workflow system, such as the large exglobal script that runs the operational GFS. This was done to allow the NEMS regression tests to be used as a test of the GFS workflow. This harms portability and adaptability by forcing users to make vast script changes to add new coupled components, or adapt scripts to a new model.

The inclusion of workflow components has another major disadvantage: it gives GFS developers a false sense of security. The rt.sh regression tests are not a sufficient test of the GFS workflow. They only test the forecast job, and not any of the initialization, post-processing, data delivery or other components. There is another project presently developing a test suite for the GFS workflow system, and that suite will replace the workflow aspect of GFS testing.

The regression test suite could be made more portable by removing the operational workflow components, and simply running the NEMS coupled model on prepared sets of inputs. That new method requires rewriting the entire rt.sh regression test suite.

Linear Test Order

Tests must be run one at a time, in a specific order, and the test suite cannot rerun specific tests that failed. This has a number of consequences which slow down the development cycle:

- It can take as long as 24 hours to run the test suite. Were the jobs to be run in parallel, the entire suite could run in under two hours.
- The test suite has no automated way to re-run specific failed tests without restarting the test system. This is important to users: when they fix a problem that broke one regression test, they want to re-run just that test to see if the problem is fixed. It is possible to re-run a specific test via a manual method: editing an “rt.conf” file and re-running rt.sh. However, some users are unaware of this feature, or find it an inconvenient way to rerun failed tests. Finding a better solution to this problem was a high priority to a number of users.
- The test suite is run by a single process, which must run for the entire duration of the suite. This is error-prone, and is also not allowed due to security restrictions.

Missing Features

The test suite lacks many desired features which are difficult to add to the existing system. A full list of desired features is included in the requirements, which make up the later sections of this document. More notable missing features include:

1. *Magnitude of differences*. If a change to a model is expected to change the simulation output, the user needs to know how much the results have changed.
2. *Multiple builds*. A model can be built in multiple ways, such as with an external physics library versus using the model’s internal physics library. The current rt.sh cannot run the same tests with multiple build options.
3. *Large tests*. The current rt.sh is limited to small test jobs, which requires low-resolution, short-lived jobs. Users have requested full-resolution, long-duration simulations which may use a significant percentage of a supercomputers’ resources for several hours.
4. *Resource usage tests*. Users have requested the test suite track the usage of cpu time, disk input and output, and other resources. A test should “fail” if it increases resource usage by too much.

Overview of Requirements

This section does not list requirements; it only summarizes what is detailed in later sections. Requirements are denoted in later section by XX-JJ-K where letters XX are the category of requirement and numbers JJ-K identifies the requirement within that category. Terminology is defined in the Terminology section. Some requirement sections contain clearly marked justification or implementation consideration information. That information does not constitute a requirement, rather it is present to assist the reader and implementer.

The framework will include sets of tests, each of which contains an input step, an execution step, a verification step and a baseline generation step. During execution, one specifies a set of tests to run, and whether to run in baseline generation or verification mode. The execution will only obtain these inputs, run filters on them, run NEMS.x, and then verify outputs or generate a baseline. It will not run any other workflow components except for archiving steps (if enabled).

The new NEMS regression tests will only test the NEMS executable; these tests will not include any aspects of the model workflow such as the GFS exglobal script, nor tests of external libraries and utilities.

Compilation - The test suite will automate compilation of the NEMS.x executable, including tracking of multiple versions of the executable.

Inputs - Each test will have a directory of prepared input data; a test will not run any aspect of a model's workflow to generate this input. Any necessary workflow will already have been run to make these inputs.

Outputs - The test suite will keep a list of expected outputs from each test. Outputs will be validated against past outputs in baseline directories. The suite will be able to make new baselines or compare against old baselines. It will support bit-for-bit comparison and more advanced comparison methods, which can be specified on a per-file basis. It will support more sophisticated results than "pass" and "fail," including comparing magnitude of changes in the model output.

Execution - the test suite will execute a set of tests with known dependencies. It will have an automated workflow manager component (based on Rocoto and ecFlow) that can run multiple tests in parallel. The workflow will automate every part of the test, including compiling NEMS.x, obtaining inputs, running NEMS.x, and validating outputs or making a new baseline.

Subsets - the test suite will allow specification of any arbitrary set of tests, and will automatically resolve dependencies, adding any tests that must be run first before the ones the user specifies.

Terminology

This section defines terms used in requirements. These are not requirements; they are definitions of terms, to clarify later requirements.

- *baseline* - *output files* from *NEMS.x*, known to be valid. These may come from a prior execution of the *test suite* or from some external source.
- *batch job* - a group of compute resources used to run a set of scripts and programs
- *batch system* - a system like MOAB, Torque, LSF and ALPS that tracks compute resources and assigns them to *batch jobs*
- *dependencies* - a test may require another test to run first, and also requires other builds of the *NEMS.x*
- *ecFlow* - an *workflow management system* that automates *batch job* submission based solely on events from those *batch jobs* . The *ecFlow* system differs from *Rocoto* primarily in that *ecFlow* has no knowledge of the status of *input files* or *output files*.
- *input files* - files input to the *NEMS.x*
- *input filter* - a simple program or script to modify input files before running *NEMS.x*
- *NEMS.x* - a compiled NEMS executable
- *NEMSAppBuilder* - a build script in the NEMS that knows how to build all model components to create *NEMS.x*
- *output files* - files output from *NEMS.x*
- *Rocoto* - a *workflow management system* used to automate submission and maintenance of *batch jobs* based on those jobs' success or failure, as well as the presence or contents of *input files* and *output files*
- *test* - an execution of *NEMS.x* off of *input files*, whose *output files* are compared to a *baseline*
- *test suite* - a group of tests and associated dependencies, *input files*, *output files* and *baselines* and a specified way of building and running *NEMS.x*.
- *workflow management system* - a system such as Rocoto or ecFlow that automates the submission and tracking of *batch jobs* in a *batch system*

Regression Test Suite Requirements

SC - Scope

SC-01 Only a Test of NEMS.x

The test suite is only a test of the NEMS executable. It will not run the workflow system for a model, nor test any external libraries and utilities. It will not test external libraries and utilities. Instead, the modeler has the responsibility to run the workflow themselves and provide inputs to the NEMS.x in a prepared directory for each test. External libraries and utilities must already be available before the test suite begins its execution.

SC-02 Documentation

The test suite will include documentation of both its internal design and how to use the suite.

TS - Test Subsets

TS-01 Any Valid Subset of the Suite Should be Runnable

Any subset of the test suite that does not violate test dependency specifications will be easily runnable.

Justification: This has been critical to debugging in the current GSM/NMMB test suite and will be critical to new models as well.

TS-02 Set Specification

TS-02-1 Test Lists

The test suite will allow the user to provide a list of tests to run, and accept this as a new set of tests.

TS-02-2 Set Arithmetic

The test suite will support simple set arithmetic operators, including union, intersection and set subtraction.

TS-02-3 Automatic Dependency Resolution

The test suite will resolve any dependencies in a set of tests automatically, and add those dependencies to the list of tests to run.

BS - Build System

BS-01 - Include Tests of Alternative Build Options

The test suite will be able to run the same execution tests with multiple executables from different build configurations.

Justification: This is needed to support different build modes, such as building with or without an external physics library.

BS-02 Common Build System

The test suite's automated build capability will use a common build system for all components.

Implementation considerations:

The only common build system in the NEMS suite right now is the *NEMSAppBuilder*, a long shell script that sits on top of individual models' build systems. However, it does not have an automated build capability. Also, it may not call the NEMS build system the same way the old regression tests did. This means either:

1. The NEMSAppBuilder will be replaced with a new system, or
2. the NEMSAppBuilder will be updated to be able to
    a. build the same way as the rt.sh build system,
    b. and have an unattended build capability.

BS-03 Build Logging

All steps of building the NEMS.x will be logged.

Implementation considerations: this may require changes to the NEMSAppBuilder and underlying component build systems.

IM - Implementation

IM-01 Based on Compsets

This new regression test suite will be based on compsets, rather than the current NEMS/tests/tests profiles test system, as the regression tests are now. This will be

accomplished by replacing the compsets' underlying connection to the regression test system with an independent implementation. The old regression tests will then be individually added to the new system as compsets.

IM-02 Usable by All NEMS Models

The NEMS system has many models, not just GSM and NMM-B. The test suite will be usable by any one model, and any coupled combination.

Justification: without this capability, it is not really the NEMS regression test suite; it is just the regression test suite for two models.

IM-03 Only Minimal Scripting Knowledge Required to Use Suite

The test suite will provide a simple front-end to a tool that lets users do the following without having to edit scripts. This is in the spirit of rt.sh and rt.conf, but may use a different syntax. It will:

1. add new tests for presently supported models or coupled combinations,
2. run the suite,
3. run a subset of the suite,
4. generate a new baseline, or
5. specifying which input pre-processor or output comparison operators to use.

Furthermore, only minimal scripting knowledge will be required to add to the tool support for:

1. A new model or coupled combination,
2. a new input file filtering method, or
3. a new output file comparison method.

Training and documentation will be provided to scientists, developers, and testers as needed to satisfy the above requirements.

EX - Execution

EX-01 Tests Will Not Require Long-Running User Processes

The test suite should not require a user process to run for the entire duration of the test suite execution. Instead, processes will only live as long as one batch job or one test.

Justification: This requirement is needed because long-running user processes are often banned due to security restrictions, are error-prone, and are vulnerable to failure due to external problems.

EX-02 Workflow Automation

The test suite will be able to automate the tests in two different ways.

Justification: This will give the user maximum flexibility to fit the test suite within the limitations of their cluster and project resources. It will also make it easier to run tests in an automated manner.

EX-02-1 Interactive Batch Job

On systems that have the capability of interactive batch logins to compute nodes, the suite will be able to run tests and report their success or failure on the local machine.

EX-02-2 Automated Workflow Manager with Unattended Execution

The test suite will be able to use automated workflow management systems, and will be portable to other workflow systems. Examples of workflow systems are Rocoto and ecFlow. Any dependencies between tests will be defined in the workflow management system so that the workflow manager can run automatically without further interaction with the user. All machines that NEMS supports will have at least one automated workflow manager option for running the NEMS regression tests.

Justification:

This allows continuation of two use cases that have proven critical:

1. Users have run weekly regression tests for the entire NEMS community.
2. Some users have run nightly regression tests to verify their own code.

Note that the frequency of these unattended runs is not a requirement in this document; that is considered outside the scope of this document.

EX-03 Rerunning Tests

If the test suite stops part way through for any reason, it will be possible to rerun tests without rerunning the entire suite.

EX-04 Execution Logging

All steps of execution will be logged.

RC - Reporting Capabilities

RC-01 Consistent and Correct

The reports from the test suite will be consistent, and any report should be correct.

RC-02 Text Output with Deterministic Order

The test suite will have the capability of creating an easily-parseable text output file listing the results of each test and other relevant information. The order of the tests in the file will be deterministic; no matter what order the tests were run, the order in the text output will be the same.

RC-03 Revision Tracking and Reporting

The revisions of repositories used to build or run the NEMS.x will be tracked and reported by the test suite.

IO - Input and Output

IO-01 Archiving

The test suite will be able to interact with an archiving system, obtaining inputs and baselines from an archive, or sending a new baseline to the archiving system.

Justification: The archiving capability ensures that data loss due to power failures will not render the test suite unusable.

IO-01-1 Checksum

Archives will have an associated checksum stored in two locations:

1. In the Subversion repository, along with information on where the archive may be found.
2. In a separate file next to the archive, so the archive may be validated outside the test suite.

Justification:

The presence of a checksum in the Subversion repository ensures the user can validate the archive against what was intended for that revision of the repository. This is because only a Subversion administrator can delete a revision from the repository; a user cannot.

Implementation note: Decisions on who updates the archives and checksums are outside of the scope of this document.

IO-02 Baseline

The test suite will be able to automatically compare outputs against a baseline, or automatically generate new baselines. The user will specify one of these two modes in which to run when starting the test suite.

IO-02-1 Automatic Baseline Generation

The test suite will have the capability of automatically creating a new baseline for any set of tests.

IO-02-2 External Baseline Sources

The test suite will be able to compare against baselines generated by some other workflow. Possible sources include, but are not limited to, a model's parallel scripts, the operational (NCO) system, or the old rt.sh test suite.

IO-02-3 Multiple Baselines

It will be possible to have multiple baselines, such as for old versions of the code.

IO-03 File Comparison Operators

The test suite will have the ability to use more than one type of file comparison method, here referred to as an operator. Each test will have a specified list of files to compare, and will specify one or more operators for each file, or group of files, as well as a default operator in cases where none are specified.

It will be easy to add a new file comparison operator to the test suite, so long as it has the ability to take two files as input and produce a yes/no answer as to whether the files are sufficiently similar. If the operator provides additional information such as a similarity metric, the test suite will propagate that information back to the user.

IO-04 Input Filters

The test suite will have the ability to add input file filters. These are operators that generate input files for NEMS.x from one or more files taken from the input directory.

IO-05 Input File Areas

The test suite will have the capability of using input and output from both of:

1. A user-specified area with user-provided data, and

2. A shared project area with maintained canned cases