Plan to transition EMC developers to CCPP

Draft: July 5, 2018. Updated multiple times; last time on Sep 30.

Table of Contents

Location and use of physics for FV3GFS

The location and use of physics will change as the code evolves toward full CCPP integration (Table 1). There will be a transition period during which all CCPP code is being integrated as IFDEFs. When compilation option CCPP=N is chosen, the original unaltered code is used. When compilation option CCPP=Y is chosen the CCPP is used either in full implementation mode (HYBRID=N) or using a dual capability to run both original and CCPP-compliant physics (HYBRID=Y, which is currently the default).

	Now	**After CCPP final integration**	**During the transition**		
			CCPP capability CCPP=Y HYBRID=N	**CCPP=N**	**CCPP hybrid** CCPP=Y HYBRID=Y
Location of	Subdirectory	ccpp-physics	No physics is	Nothing	No physics is

physics	of FV3 (gfsphysics)	repository	removed from gfsphysics, but CCPP-compliant schemes used come from CCPP repository	changes, same as now	removed from gfsphysics, but CCPP-compliant schemes used come from CCPP repository
Way to choose physics	Namelist	Suite Definition File and Namelist	Suite Definition File and Namelist	Nothing changes, same as now	Suite Definition File and Namelist
Way to invoke physics	GFS_physics_driver, GFS_radiation_driver and GFDL mp fast physics	Suite Definition File	Suite Definition File	Nothing changes, same as now	GFS_physics_driver with dual capability (ability to run both CCPP-compliant and original parameterizations in the same run). Radiation called directly through CCPP Framework (GFS_radiation_driver not used)

The full use of CCPP described in Table 1 (CCPP=Y, HYBRID=N) allows the choice of runtime parameterizations through the SDF, and is termed the FLEXIBILITY mode of the CCPP. Additionally, a PERFORMANCE model of the CCPP is available through the options STATIC=Y SUITE=xyz.xml. In the PERFORMANCE mode, the SDF must be determined at compile time and a static physics library is used, allowing the code to run faster.

Source code. There are currently two locations for physical parameterizations: the CCPP-Physics repository contains CCPP-compliant parameterizations, while subdirectory *gfsphysics* of the FV3 repository contains non-CCPP-compliant parameterizations. Once the

transition is complete, the only location for parameterizations will be the CCPP-Physics repository.

Currently parameterizations from one or the other source code locations are used depending on the type of build. When CCPP=Y is used, all CCPP-compliant schemes come from the CCPP-Physics repository; if HYBRID=Y is used, any non-CCPP-compliant schemes come from the FV3 repository. If CCPP=N is used, all schemes come from the FV3 repository.

During the transition period, parameterizations will exist in both locations and GMTB will keep the ones in the ccpp-physics repository updated when updates happen in *gfsphysics.*

Physics selection. Physics is currently selected using a namelist. Once the transition to CCPP is completed, physics selection will happen through the CCPP Suite Definition File but consistency checks will be needed with namelist settings. When the CCPP is used in FLEXIBILITY mode, parameterizations can be chosen at runtime. When the CCPP is sed in PERFORMANCE mode, parameterizations must be chosen at compilation time.

Calling Physics. Physics is currently invoked using drivers (GFS_physics_driver GFS_radiation_driver) or, in the case of GFDL microphysics fast physics, directly through subroutine calls in the Lagrangian loop within the dynamical core. Once the transition to CCPP is completed, no drivers will be necessary. All physics will be invoked through calls to the CCPP framework within the host application cap. During the transition period a capability to run both CCPP-compliant and original parameterizations will be maintained. For this purpose, the GFS_physics_driver is being modified to support this dual capability. Since all radiation parameterizations in FV3 are CCPP compliant (short- and long-wave RRTMG), this dual capability is not necessary for radiation and the GFS_radiation_driver can be eliminated during the transition phase.

Transition timeline

- May 2018: portability changes submitted and accepted to master of FV3, NEMS, and NEMSFV3GFS repositories
- June 2018: integration of CCPP framework submitted and review process started
- August 2018:
    - GFS_physics_driver dual capability to be submitted for use with updated ccpp-framework and ccpp-physics codes.
    - ccpp-physics repository updated with parameterizations of the FV3GFS default suite as of July 2018
    - Need to allow time for review by EMC and acceptance into trunk. GMTB will periodically update branch to stay up to date with master
- September 2018 (by EMC and GMTB)
    - Ability to run GFS suite with CCPP in EMC master

- September 2018 (by GMTB)
    - ccpp-physics repository updated with parameterizations of the CPT/EMC suite (SHOC, CS, MG, Ozone, H2O). October 2018 (by GMTB)
    - In person training provided to EMC staff at NCWCP
- Fall 2018 (by GSD)
    - ccpp-physics repository updated with parameterizations of the RAP/HRRR suite (GF, Thompson, RUC LSM, MYNN surface and PBL)
- Fall 2018
    - CCPP used in parallels for physics tests toward FV3GFS v2
- December 2018 (by PSD)
    - ccpp-physics repository updated with RRTMGP
- February 2019
    - Unified GWD in CCPP (by GMTB)
- January 2021
    - Operational Implementation of CCPP as part of FV3GFS v2

Criteria for transition to master

- Bit-for-bit reproducibility
    - GMTB has obtained bit-for-bit reproducibility of runs with and without CCPP when using the CCPP=Y (for all tests in rt_ccpp_hybrid.conf), as well as CCPP=N (for all tests in rt.conf) and “CCPP=Y HYBRID=N” (for all tests in rt_ccpp_standalone.conf). Reproducibility tests are performed at REPRO=Y in the config file. Differences between REPRO and PROD: REPRO is missing the following optimization flags (all others equal):
        - -no-prec-div -no-prec-sqrt
        - -xCORE-AVX2
        - -qopt-prefetch=3 (removing this may not be required)
- Pass all regression tests
    - All tests pass
- Code is stable (no crashes)
    - No crashed occurred
- Timing and memory statistics available
- Evaluation impact of changes in compiler version and flags.
    - See here preliminary results
    - Test setup:
        - For runs with CCPP, use static build
        - Theia
        - C768 resolution
        - No DA - free forecast
        - 16-day forecast

        - No CCPP: Intel 18, Intel 15
    - Test 1: Impact of compiler version for runs without CCPP
        - CCPP=N
        - Intel 18 versus Intel 15
    - Test 2: Impact of removing 3 compiler flags
        - -no-prec-div -no-prec-sqrt
        - -xCORE-AVX2
        - -qopt-prefetch=3

Criteria for transitioning CCPP to be the only way to call physics in the EMC FV3

- All necessary physical parameterizations that are in EMC's master today need to be CCPP-compliant.
    - Example: WSM6, RAS, several versions of the moninX PBL schemes, and an older version of Thompson are currently in FV3 and GFS_physics_driver. It appears there is some development in these items. If needed, they must be moved to CCPP.
    - Need plan for addition of new schemes for FV3GFS. For example, OU/CAPS would like to add some physics for the regional application. Recommendation is that any new parameterization to be introduced needs to be CCPP-compliant. If it needs to be run with non-CCPP-compliant scheme, use the dual capability implementation.
- Identify and prioritize a list of tests that need to pass and plan the work accordingly:
    - Run with fv3gfs workflow (free and cycled forecasts)
        - Involves running the ensemble DA configuration
    - Coupled model runs: must be able to build and use CCPP for relevant ocean-ice-atmosphere runs. This involves creating one or more CompSet(s).
    - Ensemble runs?? GEFS???
    - NGAQ
    - Flags and options (ldiag3d, lgocart, …?)
    - What else?

Criteria for transition of CCPP to operations

- Fully optimized performance timing and memory; similar or better performance than non-CCPP code.
- Performance tests of static versus dynamic loading of CCPP library

Code Management

The ccpp-physics and ccpp-framework are public repositories under the NCAR account in GitHub. They are included as submodules in the NEMSFV3GFS repository (could be transitioned to the *manage_externals* if this tool is adopted by NOAA for UFS management). The Governance for these codes is still under development and in transition from being held by GMTB to being held by a larger group. Therefore, the procedures listed will likely be rapidly evolving in the next few months. These are also contingent on a Hurricane Supplemental proposal (under review) for the development of unit, function, and regression tests for the ccpp-framework, along with a continuous integration strategy.

In order to submit an innovation, whether it is a bug fix, enhancement to an existing scheme, or a new feature, developers should create their own copy (github fork) of the repository, create a branch, add their innovation to the branch, make sure their branch is up to date with the master branch in the NCAR account, and submit a request to commit the code to master (pull request). All developers will receive notice of the PR and can comment. An approval by the "code owners" of the repository is necessary before a PR can be merged to the NCAR master branch (code owners are specified in a file named CODEOWNERS at the top level of the repository). Current code owners are from NCAR and NOAA GMTB, but this can be expanded in the future. Issue tracking is enabled in both the ccpp-physics and ccpp-framework repositories. Support can be requested by emailing gmtb-help@ucar.edu.

Technical approval of PRs depends on successfully passing the NEMSFV3GFS regression tests (RTs) on Theia. Regression tests have been developed for the CCPP and will be integrated to the default NEMSFV3GFS RT. The CCPP-Physics change Control Board (currently being established) will determine which parameterizations will be included in the supported public release.

Both ccpp-physics and ccpp-framework are community codes that can be used by a variety of modeling systems and institutions, including NOAA's FV3-based Unified Forecast System (UFS) and GMTB's single column model. Collaborations are being started to use the ccpp-framework in NCAR models (WRF, CESM, MPAS). This requires testing involving multiple models, but it is also important to recognize that not all developers will be able to run all tests (e.g., NOAA folks are not expected to test ccpp innovations in MPAS). A collaborative testing framework will be devised as soon as the governance is approved to include additional groups.

Training and transition to scientists

We recommend that all developers be trained in Git and GitHub development tools before receiving training specific about the CCPP code management. There are many online free resources for that.

Initially, EMC staff can compile the NEMSFV3GFS code with the option CCPP=N, which is the default. In this way, the code will not be affected by CCPP since all CCPP code is isolated through the use of preprocessing directives (that is, the CCPP code is IFDEFed out). As the transition proceeds, staff can compile with CCPP=Y, to evoke the CCPP-compliant parameterizations through the CCPP framework.

Starting in October, GMTB staff can provide training to EMC staff through any prefered method (GoToMeeting with demo or in person). Topics for training can include CCPP technical overview, overview of host model cap, how to make a parameterization CCPP-compliant, features and use of the ccpp_prebuild.py script, and CCPP code management and testing.

Known software issues

- It is not possible to check out a code from Github.com on Dell and Cray WCOSS machines. Apparently this is due to outdated versions of git and/or SSL. Jun asked wcoss system support to resolve this. Until then, the solution is to use WCOSS IBM machines (Tide and Gyre) to access GitHub. Since all WCOSS filesystems are cross-mounted, scientists can develop software on WCOSS Dell & Cray by running their git commands on Tide/Gyre. The MOM6 and ADCIRC developers are already used to this workaround.
- CCPP is untested on wcoss. Mitigation: it would be good for CCPP developers to have access to wcoss in order to test CCPP.
- NEMSFV3GFS regression test on Theia uses Intel v15, CCPP requires Intel >= 17. Mitigation: upgrade NEMSFV3GFS regression test to use more current compiler version (ongoing work by Jun Wang).
- CCPP will not run on Jet due to old software stack. Mitigation: Need Intel 17+ installed on Jet.
- Three changes are needed in NCEP Libraries, as per VLab Redmine tickets:
    - 49011: Need -fPIC in compilations so NCEP Libs can be linked against dynamic libraries (pending since April 2018 and recently assigned to Hang Lei)
    - 50320: Compile NCEP libraries with OpenMP flag on system that (can) use OpenMP. Code changes also required (pending since June 2018)
    - 50321: Code changes needed for portability (pending since May 2018)
    - Mitigation: link to prebuilt libraries that use necessary compiler options
- FMS in GFDL repository lacks certain changes required to run FV3+CCPP in all currently supported platforms. Mitigations: Submit pull requests with necessary change to the FMS GFDL repository; or use a EMC clone of FMS with needed portability changes until UFS-SC provides guidelines of which platforms should be supported and corresponding repository structure is put in place.