EUROPEAN COMMISSION

Community research

is-enes
INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

IS-ENES2 DELIVERABLE (D -N°: 3.3)

Report on common radiation code

{File name: IS-ENES2_D3.3.pdf}

Authors: **G. Hime, B. Stevens, J.L. Dufresne**

Reviewers: **James Manners, Quentin Libois**

Reporting period: ***01/04/2016 – 31/03/2017***

Release date for review: ***20/04/2017*** Final date of issue: ***19/05/2017***

Revision table			
Version	**Date**	**Name**	**Comments**
1	20/04		Version for reviewers
2	19/05		Incorporated changes requested by reviewers

Abstract

We developed a software toolkit for computing radiative transfer in the atmosphere. It is an improvement over existing libraries for the same purpose, with increased portability, modularity and maintainability. The prototype version has been successfully adopted by different research groups, each with different technical constraints and scientific requirements, for which the previously available tools were inadequate. In addition, the CFMIP Observation Simulator Package (COSP) code has been improved and optimized, and a workshop allowed the emergence of a developers and users community.

Project co-funded by the European Commission's Seventh Framework Programme (FP7; 2007-2013) under the grant agreement n°312979		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants including the Commission Services	
RE	Restricted to a group specified by the partners of the **IS-ENES2** project	
CO	Confidential, only for partners of the **IS-ENES2** project	

Table of contents

1. Objectives 4
2. Results 5
   2.1 Radiative-Transfer Toolkit 5
   2.2 COSP Network Support and Optimisation Workshop 6
3. Perspectives 8
   3.1 Radiative-Transfer Toolkit 8
   3.2 The COSP Community 9

Executive Summary

This deliverable concerns the development of a set of common radiation tools and the concomitant creation of a user and developer community for it. These radiation tools consist in a common radiative transfer library to be used in Earth System Models, and the communities' work on observation-system simulators. A prototype has been developed through the refactoring of the existing and widely adopted RRTM developed and maintained by the Atmospheric and Environmental Research since the 1980's, allowing for relatively smooth migration through its backward compatibility. The new code compiles independently from any particular GCM, and can therefore be built as an offline, radiation-only tool. This provides the scientific community with a light-weight development sandbox, in which further use and development is not burdened by the weight of a whole GCM. This radiative transfer library can be used quite immediately through C/C++ bindings and a Python wrapper; the latter allows an easy, interactive interface. Additionally, a new ocean surface albedo code has been developed to be more consistent with new observation data and new Earth System Models specificities such as computing marine biota and detrital organic materials. Finally, the COSP (CFMIP Observation Simulator Package) code has been improved, optimized and integrated to all European climate models, and a workshop has formalized the emergence of a COSP developer and user community. Work on this deliverable has continued without interruption after the end of the IS-ENES2 calendar, and the parties involved have already committed themselves and secured funding for further development in the next three years.

1. Objectives

Several weather and climate models, such as the MPI-M's ECHAM and ICON, rely on the Atmospheric and Environmental Research (AER, www.aer.com) RRTM (Rapid Radiative Transfer Model, www.rtweb.aer.com), publicly available under a limited license for non-commercial use. This code embodies several decades of experience and consolidates data and formulae from different sources. It has become a *de facto* standard against which alternative libraries for radiative transfer can be compared. It is written in Fortran 77 and has had its last major rewrite (RRTM v. 3, still in Fortran 77) in 2002. A major rewrite named RRTMGP is currently under development, this time in Fortran 2008, i.e. the newest language standard.

However, each GCM that has adopted the original RRTM has integrated the original library into its own codebase, i.e. the original code gradually diverged from the AER version. This means changes made by AER since RRTM was first integrated into a given model may never make their way into said model, because changes were made in the already embedded version (e.g. for compatibility or experimentation purposes) prevent the immediate adoption of a new AER release. As the AER's new releases and the embedded versions of RRTM diverge, results produce by them diverge as well. Since these models are production software, be it scientific or weather forecast production, a qualitative disruption in the model output is to be avoided at all costs. The only way to keep consistency throughout the transition to a new radiative transfer model is to perform incremental modifications in the existing and hard-integrated version of RRTM already found in e.g. ICON. Accordingly, the development framework is that of ICON itself, namely Linux, Fortran and Git.

An incremental work plan was produced after a few months of preliminary assessment, in which each deliverable represents significant gains for the scientific community. The main thread of this plan is the reengineering of the existing radiation library in ICON, minimizing disruptions in the current workflow that currently relies on it and continuously increasing its functionality and performance along the way. ICON and its embedded version of RRTM were chosen as starting point for practical reasons: the work was carried out at the MPI-M and ICON has a large user and developed community already, making for a tight feedback loop and immediate deployment of partial results. The longer-term goal is to expand this community to absorb other research and weather forecast communities as the quality of the radiative transfer developed as of IS-ENES2 increases.

Finally, a workshop for the consolidation of the already established and ongoing COSP project was to take place, to assess its progress and outline its future.

2. Results

2.1 Radiative-Transfer Toolkit

The first milestone was the removal of the embedded version of RRTM from the entangled network of dependencies in the source tree of ICON, effectively returning the highly integrated version of RRTM inside the GCM to a state that can be considered a library. The result of this work was a code which compiles independently from the model and can therefore be used to build an offline, radiation-only tool, but still is 100% backward compatible and generates the same results to which ICON has been calibrated. This allows the user community to pin-point the divergences as they are made evident during the refactoring process. Moreover, the large amount of expertise accumulated by the equally large user base of ICON provides a safe background against which regression testing can be performed.

A corollary benefit from this work was that now the code can be used as a light-weight development sandbox, in which further compilation and debugging of the code is not burdened by the weight of a whole GCM. The radiative transfer code of ICON can now be used quite immediately through C/C++ bindings, through which all programming languages can interface; in particular, we have already written a Python wrapper for said interfaces, which is already being used by a small but growing number of scientists and PhD candidates. Also from this point, it was already possible to share the library with another GCM – case in point being the IPSL's LMDZ. The integration was carried out between June and August of 2016 at LMD/IPSL, where the library calls were coupled to the data structures inside the GCM. The results were superior to those generated by the previous, over-simplified model that was in use before, to the point of being qualitatively different. Further assessment is required at the LMD to ascertain the possibilities opened by the integration of this new radiative transfer library, but in regard to the scope of this project the results can be considered a major success.

The refactoring work has continued throughout the remaining duration of the project and beyond. It was first backmerged into the MPI-M main development branch in late March 2017. As of the writing of this report, the current Software Engineering metrics can be summarized by a reduction of 75% in number of effective lines of code. This is due to the conflation of redundant data structures and routines, e.g. the refactoring of similar routines with hard-coded data into data-driven ones. The current code is numerically and computationally equivalent to the original one, only much smaller, readable and flexible. Even if one assumes the readability, flexibility and other arguably subjective qualities of the code had remained unchanged, which is not the case, the sheer decrease in size already represents a reduction in the cost of further developments by a factor of four. Moreover, despite no optimization for performance having been conducted so far - as this ground-work must needs be completed first - the new code requires half as much memory, produces half as many cache misses and, depending on the test-case, can perform twice as fast.

In parallel, because of the delays in the development of the new radiative transfer library, LMD put efforts in the development of a new accurate ocean surface albedo model consistent with the requirements of the next generation of Earth System Models (ESMs). Most of the schemes currently used in ESMs do not resolve spectral variations in ocean surface albedo (OSA) thus excluding subtle processes. More specifically, the spatial-temporal variability of foam and ocean color in response to both fluctuations and long-term changes in climate cause changes in the ocean surface albedo; this in turn impacts the amount of solar radiation absorbed by the upper layers of the ocean, and from that the stability of the ocean mixed layer, the sea surface temperature, and a multitude of biophysical and biogeochemical processes.

The Solar zenith angle (SZA) is the single most prominent driving parameter for OSA; other parameters that affect OSA such are the repartitions of incoming solar radiation between its direct and diffuse components, the sea surface state (often approximated through the surface wind), the concentration of suspended matter and plankton light-sensitive pigment, and white caps. We developed an ocean surface albedo scheme that includes these relevant processes based on the literature published over the last years. This spectral scheme resolves the various contributions of the surface for direct and diffuse shortwave radiation. This scheme has been implemented in two Earth system models (IPSL-CM and CNRM-CM). It has been compared with satellite data and available *in situ* observations Results show a substantial improvement in the simulated OSA, when compared to previous scheme. A publication is in preparation. This new code will be implemented in the new radiative framework with institute internal funding.

2.2 COSP Network Support and Optimisation Workshop

The workshop took place at UPMC, Paris, 27-28 February 2017 and was attended by 21 participants involved in the development of COSP or in its implementation in climate models. Six climate modelling groups from Japan, USA, and Europe were represented: EC-EARTH, GFDL, IPSL, MIROC, MPI-M, and UM. This was the first workshop of this kind. It considered a success, and one of the recommendations is that it should take place every two years. The workshop covered three main topics in three different sessions (see agenda attached). The main outcomes of the discussion sessions are summarized below.

Session 1: Current implementations

- **Documentation.** There was strong agreement that the documentation needs to be updated and expanded. The preference is to use the capabilities provided by GitHub so that this documentation can be easily updated and the changes traceable (wiki, web pages with version control). The efforts should be focused on the following aspects.

- Implementation guide. This section aims to help model developers with the implementation of COSP (see also section below on debugging). Of particular importance is the addition of more stringent tests that can capture errors in the implementation. These tests must provide an interface for specifying the required microphysical configuration. The outputs should match the plots generated with known good outputs.
- Configuration guide. Once the COSP has been implemented, the developer should know which things need to be changed to make the implementation consistent with the host model. The guide should provide practical information on, among others, microphysical configuration, setting the number of sub-columns,replacement of the sub-column cloud generator; proper handling of missing data, description of diagnostics that need averaging masks in models that don't handle missing data values and grouping those diagnostics that share the same mask.
- Good practices. COSP aims to facilitate comparison between model outputs and satellite or *in situ* observations. However, users must be aware of limitations inherent to such comparisons. For instance, the horizontal resolution of models may be better than that of observations, and adequate interpolation methods should be used. Another example is the radar reflectivity for which the shape of the frequency by altitude histogram is more relevant that the exact values.
- Frequently Asked Questions section.

- **External library.** Provide the capability of building COSP as an external library to facilitate in-line implementation. COSP2 is already built in two stages, so this should be easy to do.
- Share technical information on implementation in models. There has been very little exchange of technical information about how COSP is implemented in different models. If the implementations differ substantially from the 'canonical implementation' used in the cosp_test driver programs, then the maintenance cost increases substantially (large human effort to keep inline version up-to-date with the COSP trunk).
- Optimisation. The radar simulator is still very expensive for routine use.
- Post-processing tools. If the COSP outputs need post-processing to make them comparable to the CFMIP-OBS, the post-processing scripts should be made available in the COSP repository. They will be managed using the same working practices as the COSP Fortran code. One example of this is the post-processing required to compare COSP/PARASOL diagnostics with the PARASOL CFMIP-OBS.
- Debugging. The COSP quicklooks are useful, but they do not provide enough information on the validity of the implementation. Additional diagnostics to verify and debug a COSP implementation would be helpful. These are examples of diagnostics that could help.
  - Cross-simulator checks. Some outputs in different simulators are tightly related, and therefore can be used as consistency checks.

- Sub-column diagnostics.
- Subroutine that fills in the inputs with a predetermined input data so that the in-line implementation is tested with the same inputs in every gridbox.

Session 2: Software development working practices

- Development practices needed. The PMC working practices document (attached) was well received. There was consensus that this is needed to facilitate community contributions.
- Simple coding guidelines. Simple, but clear coding guidelines are required. They do not necessarily have to focus on coding style, but on aspects like: intent declaration, banned instructions (e.g. stop statements), maximum line width, etc.
- Travis automatic unit testing. The use of this technology is highly recommended, but only if a good Fortran compiler can be used.
- Improvement of the robustness of the testing. Clear performance benchmarks are needed, and any modification that reduces the performance and is not protected by a logical switch will have to be strongly motivated (this has to be included in the working practices). The development of unit tests for each simulator is highly desirable. Due to numerical errors, models can provide non-physical values (e.g. slightly negative cloud fraction or liquid water content instead of null values) and the simulator should be protected again these non-physical values if they occur. Some input files or methods to test this problem would be useful

3. Perspectives

3.1 Radiative-Transfer Toolkit

We have seen a great affluence of partners and interest in the developments done over the course of this project, so far including the DWD, MPI-M, CSCS/MeteoSwiss/ETH and IPSL. In spite of the delays and divergence from the original work plan, the modest person-month investment successfully produced a prototype radiation library to which several European models are being ported too, and which is simultaneously being used by multiple university groups, through its Python bindings, as a standalone tool. This code continues to be developed and documented through MPI-M and CSCS internal funding (in association with the Swiss-funded ENIAC project), and is the basis for parallel efforts centered around work on computational kernels for performance portability (liking to the EU funded ESCAPE project) and exploration of asynchronous methods of radiative transfer in the spirit of process concurrency. As of today, tasks to be performed with some overlapping include: performance assessments and general optimization, architecture specific optimizations (including a GPU version), reverse engineering and documentation of the computations carried out by the

library, cross-testing against the other available packages for radiative-transfer, i.e. those assessed at the beginning of the project, integration into scientific and weather-forecast production models. Incremental releases with these features are expected to take place over the following years, as funding for continued, long-term work has been secured by the MPI-M and the CSCS. A public release of the radiative library is expected to be made later this year, under an open license for scientific purposes.

3.2 The COSP Community

To further consolidate working practices, formal engagement with the PMC will be required before new developments aimed to be merged into the trunk will be required. Owners should commit to long-term support. In addition, although the creation of compatible observations is not strictly required to add new capabilities to COSP, it is highly desirable to facilitate the use of the new capabilities. CMORisation and inclusion in Obs4MIP of some observational datasets is still difficult. IPSL has kindly made the CFMIP-OBS server available to host new COSP-compatible observations. Even if the observations cannot meet the Obs4MIPs standards, it is highly recommended to try and make them CF-compatible to facilitate their used by some standard software packages.
There is a need for customized visualization tools. Two main classes of tools are required:

- Model checks. These will be routines that help model developers to check that the COSP implementation is correct. These are considered important development tools and should follow the same standard practices as the COSP code. They will live in the COSP repository.
- General tools. These are useful tools that are not critical for the development of COSP, and don’t need support or a review process. They don’t necessarily have to live in the COSP repository. Other options like the CFMIP diagnostics repository should be considered.

There was consensus that the workshop was useful, and that it should occur regularly. A bi-annual frequency seems a good compromise. It should not be part of the CFMIP annual meeting. Lastly, it was recommended that modelling centres publicise which runs and COSP diagnostics they will be producing for CFMIP3.