Environmental Modelling & Software 26 (2011) 680–682

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Software, Data and Modelling News

OOF$_\varepsilon$: A Python engine for automating regional and coastal ocean forecasts☆

Martinho Marta-Almeida [a,*], Manuel Ruiz-Villarreal [b], Pablo Otero [b], Marcos Cobas [b], Alvaro Peliz [c], Rita Nolasco [a], Mauro Cirano [d], Janini Pereira [d]

[a] Centro de Estudos do Ambiente e do Mar, Universidade de Aveiro, Campus de Santiago, 3810-193 Aveiro, Portugal
[b] Instituto Español de Oceanografía, IEO, Centro Oceanográfico A Coruña, Muelle de Animas s/n, 15001 A Coruña, Galicia, Spain
[c] Instituto de Oceanografia, Universidade de Lisboa, 1749-016 Lisboa, Portugal
[d] Departamento de Física da Terra e do Meio Ambiente, Universidade Federal da Bahia, 40170-280 Salvador, Bahia, Brazil

ARTICLE INFO

Article history:
Received 17 May 2010
Received in revised form
21 October 2010
Accepted 27 November 2010
Available online 31 December 2010

Keywords:
Operational modelling
Ocean forecast
Python
ROMS

ABSTRACT

Coastal and regional ocean forecasts can be currently performed on a daily basis due to the advances in numerical techniques and in computational resources. Maintenance of routine forecasts is a demanding task from the point of view of software engineering since it involves a number of new additional tasks difficult to code efficiently in the compiled languages in which ocean models are written. In this contribution, we present a set of free, open-source, portable and fast modules named OOF$_\varepsilon$ – Operational Ocean Forecast Python Engine that provide a way to cope with the demanding requirements of routine execution of a regional ocean model written in a compiled language (namely the Regional Ocean Modelling System, ROMS, developed in Fortran) and that make the forecast process possible and fully automatic and robust.

1. Introduction

Advances in numerical modelling in coastal oceanography and the availability of computing power at an affordable cost are boosting the area of coastal and regional ocean prediction. State-of-the art numerical models with realistic forcing and detailed physics can be executed on a daily basis, and the scientific community is now able to investigate processes at spatial and temporal scales that could not be routinely tackled so far. However, despite affordable, the daily execution of a regional model is a demanding task from the point of view of software engineering, involving complex operations with large data files. A great number of automated tasks have to be coded for launching model simulation, gathering forcing data, storing results, plotting and visualising and afterwards for disseminating numerical output and controlling the quality of the forecast against available real-time data.

Additionally, the fact that ocean models are written in a compiled language (usually Fortran) has limitations for controlling execution and maintaining robustness and efficiency. Modern software engineering is recognising that the combination of different program languages in "mixed language" modelling packages has advantages in terms of robustness and efficiency in code development, and additionally makes it possible to address complex demanding problems in environmental modelling hard to tackle with the existing software in compiled languages (e.g. Lin, 2008; Roberts et al., 2010)

2. Why Python?

Python is an interpreted language that can be run in interactive mode, which facilitates coding for scientific applications that require use of large datasets and complex computational programs (e.g. Oliphant, 2007). Additionally, Python includes a very extensive sort of built-in modules, and consequently it can be used to run all kind of daily tasks which usually require more than one language. The base toolbox for scientific computing in Python, NumPy-SciPy, introduces numerical arrays in Python providing math libraries for data manipulation and makes possible the creation of C/Fortranextensions in an easy straightforward way. Many Python community modules of application in environmental sciences and computer science are available. We can highlight the following for their relevance in ocean modelling: i) tools to manipulate the standard data formats in ocean and atmospheric sciences (GRIB and NetCDF); ii) support for OPeNDAP; iii) tidal harmonic analysis software; iv) tools for scientific visualisation (in particular matplotlib). In this manner, the Python language allows to create and

☆ Code and documentation available at http://code.google.com/p/oofe.

* Corresponding author.
E-mail address: mma@ua.pt (M. Marta-Almeida).

1364-8152/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2010.11.015

operate the model input/output and to control all the required tasks for the operationality of the ocean model.

3. The ocean model

The operational engine is directed to the ocean model ROMS (Shchepetkin and McWilliams, 2005; Haidvogel et al., 2008) and its nesting enabled version ROMS-AGRIF (Penven et al., 2006). ROMS is a state-of-the-art free-surface terrain-following primitive equation hydrostatic model, configurable for realistic regional applications. As input data, ROMS requires grid variables, initial ocean conditions, boundary conditions and ocean forcing (like tides, river outflow, atmospheric surface heat and momentum fluxes). It is also required input text non data files, where main model parameters and other informations are specified (like the initial position of Lagrangian floats, Eulerian stations, etc, depending on the modules in use). As main output, ROMS stores snapshots and/or averages of the ocean state variables. The model also stores a restart file, which is important in the operational context since it can be used as initial condition of the subsequent model runs.

4. The OOF_{ε} engine

The operational engine is divided into two parts, one for analysis, which runs the ocean model forced with atmospheric analysis or real data, and other for forecast which runs forced with atmospheric predictions. The main mission of the Python engine is to keep the analysis and forecast models running, checking the availability of initial conditions and atmospheric forcing and verifying the success of the simulations. Each analysis and forecast cycle has 4 steps: 1) check/wait for the required forcing data and initial conditions; 2) creation of the model input files; 3) simulation; 4) plot/manage the model inputs and outputs. This cycle is repeated continuously and only stops in case of some unexpected error or missing data or file. Both ocean analysis and forecast initial conditions are obtained from the restart file of the previous day analysis.

The maintenance of the analysis and forecast cycles is performed by the main Python module of the OOF_{ε} engine (op_main). This module requires a configuration text file, where data paths, file names, characteristics of the submission job file (when using a cluster's queuing system), etc, can be defined.

Besides the main module, the engine includes three additional modules, one for the visualisation of the input/output – op_vis, another for files management of data files and graphics – op_clean, and one for monitoring the current and past ocean simulations – op_monitor.

The visualisation module can generate graphics for a variety of slices and plots. A configuration text file associated to this module allows the selection of the slices, the variables to be used, colours, etc.

The cleaning module is used to delete/compress the engine input/output files. The model input/output files and plots can easily amount to many gigabytes per day, becoming necessary to choose which files are to be deleted or compressed, a task that can be specified in a text configuration file.

The last module of the Python engine is the monitoring module. It allows to access the status of the analysis and forecast ocean model runs, namely to check if the models are running, the remaining computational time, etc.

These four modules are built on a large collection of Python tools which execute all sort of tasks involved in ocean modelling and data manipulation, like downloading data files from external servers, interpolating and extrapolating fields, creating model inputs, etc.

5. Outlook and future developments

The several analysis and plotting desktop environments in use by the oceanographic community (being Matlab the most widely used) have proved to be adequate mainly for interactive work and for setting up seasonal simulations with the ROMS model (Penven et al., 2008). These environments can be used for performing some of the tasks required in coastal model forecasts. However, the use of such tools in routine daily forecasts reveals the lack of flexibility and optimisation in terms of speed and memory use, and the non-robustness for batch mode work.

Python is a high-level, object-oriented, flexible and relatively easy to learn and use language, and consequently the maintenance of the routine execution can be performed by scientists or other operators that are not necessarily efficient code developers. Enhancements of OOF_{ε} capabilities are thus accessible and painless. Python modules can integrate all required demanding tasks including control of execution in a common framework.

The ocean model ROMS includes several modules for phenomena of interest in regional configurations, like sediment transport or biological processes, which can be handled by relatively simple extensions of OOF_{ε}. Since routines in other programming languages like Fortran can be efficiently controlled by Python scripts, existing open-source software for coupling to a wave model or an atmospheric model like the Model Coupling Toolkit (Warner et al., 2008) can be straightforwardly integrated in the OOF_{ε} platform. In the near future, it is intended to incorporate ocean state estimation through data assimilation in OOF_{ε}. Among other envisaged OOF_{ε} capabilities we can mention the development of tools for model skill assessment and quality analysis of model input and output and the incorporation of scripts for distribution of model output in an OPeNDAP server.

In summary, OOF_{ε} constitutes a powerful and efficient tool for setting up and controlling routine forecasts. In spite of being written for the model ROMS, since many others ocean models use similar input/output schemes and sequences, OOF_{ε} engine can be adapted to steer the execution and analysis of other ocean models with little effort. Operational forecast systems already implemented with OOF_{ε} are visible at the sites http://neptuno.fis.ua.pt/oof and http://oceano.fis.ufba.br/oof.

Finally, our software illustrates how the combination of new routines in the high-level scripting Python language with an existing state-of-the-art ocean model written in a compiled language has advantages in terms of code simplicity, reliability and ease-of-use, similarly to what has been reported by Lin (2008) in a Python implementation of an atmospheric model. In this contribution, additionally we show that this software approach makes it possible to tackle a new application like the maintenance of routine forecasts and cope with the demanding software requirements associated with it.

Acknowledgements

The authors acknowledge financing from EU project ECOOP (FP6 Contract No. 36355) and from project REMO (Rede de Modelagem e Observação Oceanográfica), funded by Petrobras.

References

Haidvogel, D., Arango, H., Budgell, W., Cornuelle, B., Curchitser, E., Lorenzo, E.D., Fennel, K., Geyer, W., Hermann, A., Lanerolle, L., Levin, J., McWilliams, J., Miller, A., Moore, A., Powell, T., Shchepetkin, A., Sherwood, C., Signell, R., Warner, J., Wilkin, J., 2008. Ocean forecasting interrain-following coordinates: formulation and skill assessment of the Regional Ocean Modeling System. J. Comput. Phys. 227, 3595–3624.

Lin, J.W.-B., 2008. qtcm 0.1.2: a Python implementation of the Neelin-Zeng quasi-equilibrium tropical circulation model. Geosci. Model. Dev. 1, 315–344.

Oliphant, T.E., 2007. Python for scientific computing. Comput. Sci. Eng. 9, 10–20.

Penven, P., Debreu, L., Marchesiello, P., McWilliams, J.C., 2006. Evaluation and application of the ROMS 1-way embedding procedure to the central California upwelling system. Ocean Model. 12, 157.
Penven, P., Marchesiello, P., Debreu, L., Lefvre, J., 2008. Software tools for pre- and post-processing of oceanic regional simulations. Environ. Model. Software 23, 660–662.
Roberts, J.J., Best, B.D., Dunn, D.C., Treml, E.A., Halpin, P.N., 2010. Marine geospatial ecology tools: an integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++. Environ. Model. Software 25, 1197–1207.
Shchepetkin, A.F., McWilliams, J.C., 2005. The Regional Ocean Modeling System (ROMS): a split-explicit, free-surface, topography-following coordinates ocean model. Ocean Model. 9, 347–404.
Warner, J.C., Perlin, N., Skyllingstad, E.D., 2008. Using the model coupling toolkit to couple earth system models. Environ. Model. Software 23, 1240–1249.