OceanSpy: A Python package to facilitate ocean model data analysis and visualization

June 3, 2019

Statement of Need

Simulations of ocean currents using numerical circulation models are becoming increasingly realistic. At the same time, these models generate increasingly large volumes of model output data. These trends make analysis of the model data harder for two reasons. First, researchers must use high-performance data-analysis clusters to access these large data sets. Second, they must post-process the data to extract oceanographically-useful information. Moreover, the increasing model realism encourages researchers to compare simulations to observations of the natural ocean. To achieve this task model data must be analyzed in the way observational oceanographers analyze field measurements; and, ideally, by the observational oceanographers themselves. The OceanSpy package addresses these needs.

Summary

OceanSpy is an open-source and user-friendly Python package that enables scientists and interested amateurs to analyze and visualize oceanographic data sets. OceanSpy builds on software packages developed by the Pangeo community, in particular Xarray (Hoyer and Hamman 2017), Dask (Dask Development Team 2016), and Xgcm ("Xgcm: General Circulation Model Postprocessing with Xarray," n.d.). The integration of Dask facilitates scalability, which is important for the petabyte-scale simulations that are becoming available. OceanSpy can be used as a standalone package for analysis of local circulation model output, or it can be run on a remote data-analysis cluster, such as the Johns Hopkins University SciServer system (Medvedev, Lemson, and Rippin 2016), which hosts several simulations and is publicly available. OceanSpy enables extraction, processing, and visualization of model data to (i) compare with oceanographic observations, and (ii) portray the kinematic and dynamic space-time properties of the circulation.

Features

Extraction of oceanographic properties

OceanSpy can extract information from the model data at user-defined points, along synthetic ship 'surveys', or at synthetic 'mooring arrays'. Model fields, such as, temperature, salinity, and velocity, can be extracted at arbitrary locations in the model 4D space. Thus, simulations can be compared with observations from Lagrangian (drifting) instruments in the ocean. The 'survey' extraction mode mimics a sequence of arbitrary hydrographic 'stations' (vertical profiles) connected by great-circle paths. The data on the vertical profiles are interpolated from the regular model grid onto the 'station' locations. The 'mooring array' mimics a set of oceanographic moorings at arbitrary locations. It differs from a 'survey' because data is extracted on the native model grid. This mode enables exact calculation of the model material fluxes through an arbitrary curve in latitude/longitude space, for example.

Computation of useful diagnostics

OceanSpy can compute new diagnostics that are not part of the model output. These diagnostics include vector calculus and oceanographic quantities, as shown in Table 1. For example, OceanSpy can calculate the Ertel potential vorticity field and the component of the velocity vector perpendicular to a 'survey' section. In addition, OceanSpy can calculate volume-weighted averages. When the required model output fields are available, it can also calculate heat and salt budget terms to machine precision.

Table 1: OceanSpy diagnostics. The vector velocity field is $\mathbf{u} = (u, v, w)$, which is written (for convenience) as a function of Cartesian position $x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$; χ is an arbitrary scalar field; seawater density is ρ, a function of salinity S, temperature θ, and pressure; σ_θ is the potential density anomaly; ϵ_{nh} is the non-hydrostatic parameter, which is 0 for a hydrostatic and 1 for a non-hydrostatic model; the overline denotes a time average; and the Coriolis parameter has magnitude (f, e) in the $(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ directions. Subscript H indicates a vector in the 2D $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ plane. See, for instance, (Klinger and Haine 2019) for further information.

Diagnostic name	Description
Gradient	$\nabla\chi = \frac{\partial\chi}{\partial x}\hat{\mathbf{x}} + \frac{\partial\chi}{\partial y}\hat{\mathbf{y}} + \frac{\partial\chi}{\partial z}\hat{\mathbf{z}}$
Divergence	$\nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$
Curl	$\nabla \times \mathbf{F} = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\hat{\mathbf{x}} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\hat{\mathbf{y}} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\hat{\mathbf{z}}$

Diagnostic name	Description
Scalar Laplacian	$\nabla^2\chi = \nabla \cdot \nabla\chi$
Integral	$I = \int \cdots \int \chi \, dx_1 \cdots dx_n$
Potential density anomaly	$\sigma_\theta = \rho\left(S, \theta, \text{pressure} = 0 \text{ db}\right) - 1000 \text{ kgm}^{-3}$
Brunt-Väisälä frequency	$N = \left(-\frac{g}{\rho_0}\frac{\partial \sigma_\theta}{\partial z}\right)^{1/2}$
Velocity magnitude	$\|\|\mathbf{u}\|\| = \left(u^2 + v^2 + w^2\right)^{1/2}$
Relative vorticity	$\omega = (\zeta_H, \zeta) = \nabla \times \mathbf{u}$
Kinetic energy	$KE = \frac{1}{2}\left(u^2 + v^2 + \epsilon_{nh} w^2\right)$
Eddy kinetic energy	$EKE = \frac{1}{2}\left[(u-\overline{u})^2 + (v-\overline{v})^2 + \epsilon_{nh}(w-\overline{w})^2\right]$
Horizontal divergence	$\nabla_H \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$
Horizontal shear strain	$S_s = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}$
Horizontal normal strain	$S_n = \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}$
Okubo-Weiss parameter	$OW = S_n^2 + S_s^2 - \zeta^2$
Ertel potential vorticity	$Q = -\frac{\omega \cdot \nabla\rho}{\rho} = (f+\zeta)\frac{N^2}{g} + \frac{(\zeta_H + e\hat{\mathbf{y}}) \cdot \nabla_H \rho}{\rho_0}$

Easy visualization

OceanSpy interfaces with matplotlib and xarray plotting functions and customizes them for oceanography. The most common visualizations, such as a temperature/salinity (T/S) diagrams, maps of the sea-surface temperature, or hydrographic transects along 'survey' sections, can be made with a single command. A minor change to the syntax creates an animation.

An oceanographic example: The Kögur section

Consider a specific application of OceanSpy. The Kögur section is a frequently-occupied hydrographic transect between Iceland and Greenland. It has also been instrumented by moorings for at least a year (Figure 1a). A typical task concerns comparing simulation data to these observations, for example to quantify the simulation realism and to understand how the sparse measurements represent (and distort) the 4D fields. Using the 'mooring' and 'survey' functionality of OceanSpy, one easily samples the model output on the Kögur section, computes and visualizes the velocity field orthogonal to the section (Figure 1b), computes a time series of the volume flux of dense water ($\sigma_\theta \geq 27.8$ kgm^{-3}, which selects

water that subsequently overflows through the Denmark Strait downstream of the Kögur section, Figure 1c), and explores the T/S properties of, for instance, the velocity orthogonal to the section (Figure 1d).

Relation to ongoing research projects

OceanSpy is part of an ongoing effort to democratize large numerical ocean simulation data sets, which is funded through NSF (#1835640: Collaborative Research: Framework: Data: Toward Exascale Community Ocean Circulation Modeling).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number OAC-1835640, OCE-1633124, and OCE-1756863 and by the Institute for Data Intensive Engineering and Science at John Hopkins University. The authors thank Aleksi Nummelin and Ryan Abernathey for providing constructive comments that helped us improve the code, and Salvatore Palena for designing the OceanSpy logo.

References

Dask Development Team. 2016. *Dask: Library for Dynamic Task Scheduling*. https://dask.org.

Hoyer, S., and J. Hamman. 2017. "Xarray: N-D Labeled Arrays and Datasets in Python." *Journal of Open Research Software* 5 (1). https://doi.org/10.5334/jors.148.

Klinger, B. A., and T. W. N. Haine. 2019. *Ocean Circulation in Three Dimensions*. 1st ed. cup. http://www.cambridge.org/9780521768436.

Medvedev, Dmitry, Gerard Lemson, and Mike Rippin. 2016. "SciServer Compute: Bringing Analysis Close to the Data." In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, 27:1–27:4. SSDBM '16. New York, NY, USA: ACM. https://doi.org/10.1145/2949689.2949700.

"Xgcm: General Circulation Model Postprocessing with Xarray." n.d. https://github.com/xgcm/xgcm/blob/master/doc/index.rst.

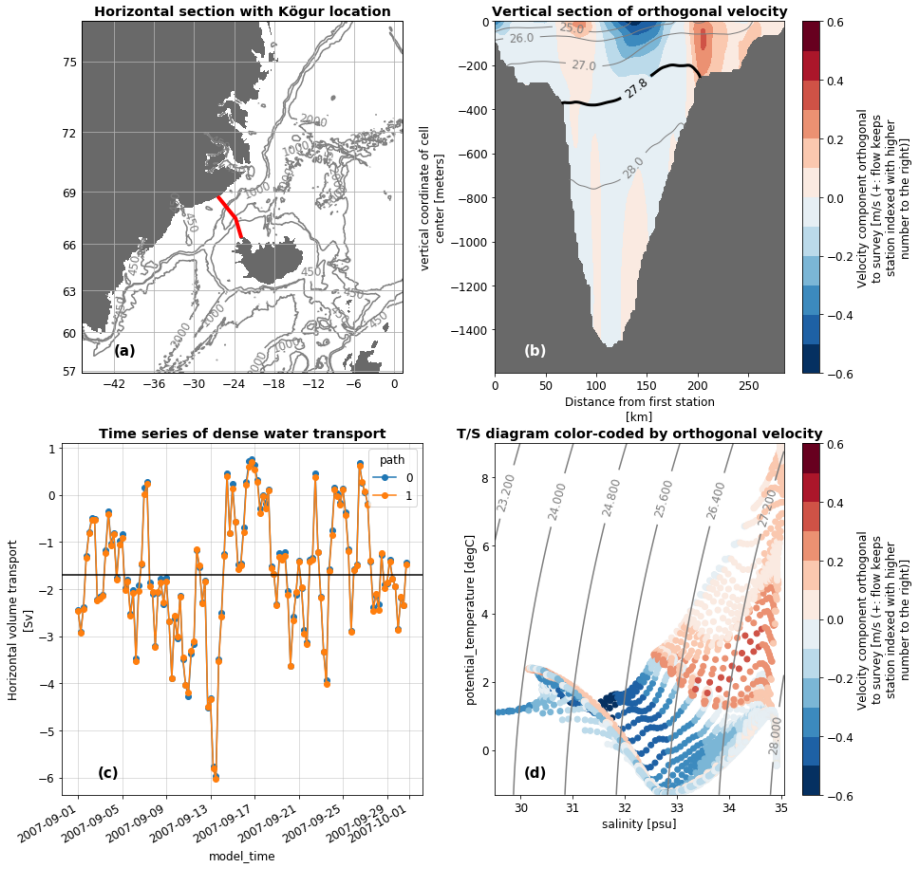

Figure 1: Extracted information for September 2007 on the Kögur section. (a) Location of the section (red line) and sea floor topography; (b) time-mean horizontal current orthogonal to the section (positive values towards the northeast); (c) volume transport (flux, in Sv=10^6 m^3s^{-1}) of dense water ($\sigma_\theta \geq 27.8$ kgm^{-3}) through the section computed following the two possible paths, with a mean southward transport of 1.69 Sv (black line); (d) T/S diagram, colored by orthogonal velocity, which shows the relatively warm salty water travels northeast, whereas the cold fresh water travels southwest.