eCSE Final Report for ARCHER2-eCSE04-7

Technical Report

eCSE ID:	ARCHER2-eCSE04-7
eCSE Title:	Porting OptClim Optimisation system to ARCHER2
PI Name:	Prof Simon Tett
Author of this document:	Dr Mike Mineter, Dr Magnus Hagdorn, Prof Simon Tett, Dr Andrew Schurer
Name of technical staff on the project:	Dr Mike Mineter, Dr Magnus Hagdorn
List of names to acknowledge with the project (by default we will use the PI, Co-Is and Technical staff members):	Dr Mike Mineter, Dr Magnus Hagdorn, Prof Simon Tett, Dr Andrew Schurer, Dr Coralia Cartis, Dr Dan Jones, Dr Maria Val Martin,
Project Start Date:	07/07/2021
Project End Date:	30/06/2022
Number of Project Months Funded:	5

Abstract

We have made two software systems available on ARCHER2 to enhance the ways in which researchers can respond to uncertainty in models. We have ported the OptClim2 software to permit researchers to optimise parameters in three Earth system models: CESM, MITgcm-based models and UKESM. We have developed "particle filtering" (PF) software for UKESM to allow an ensemble of runs to track historical data. The OptClim2 and PF systems are currently being trialled with UKESM; OptClim2 is also available for first research use with MITgcm or CESM. The development for UKESM was complicated by the need to communicate from ARCHER2 to the NCAS-provided server that runs UKESM workflow. To achieve this we used GeosMeta, which is an Edinburgh developed service providing a RESTful interface to a Mongo Database.

Introduction

Climate and Earth system models do not have perfect information, nor perfect representation of physical processes as they cannot represent all physical scales from µm to Mm. Consequently they cannot be expected to be produce exact representations of reality. When used to reproduce observations (rather than for in silico experiments to gain understanding) discrepancy can come in many forms. This includes model error either due to missing or incorrectly modelled processes, and uncertainty in small scale processes leads to large scale system uncertainty. Also any single model simulation will not have the same realisation in time of natural variability (such as simulating El Nino events).

In this project we make available on ARCHER2 two responses to this by providing software frameworks used to orchestrate runs of models:

1. OptClim provides a method for improving parametrisations in models. Models that use parametrised processes rather than detailed data and equations are

commonplace. Among examples in atmospheric modelling are those representing cloud processes – their formation and processes concerning both rain and reflection/radiation. Parameters have ranges of possible values, and with OptClim these parameters can be calibrated so models better represent past observations, and so can be projected forward with more confidence – e.g. to understand the impacts of the increasing CO2 concentrations on climate.

2. With “particle filtering” we run an ensemble of models, each differing though a random perturbation of its initial conditions. We seek to track a time-series of data, which can be historical or using the model to explore an imagined future, using a storyline approach. We can infer from the models' output data whether a particular process or change in circulation state enables the simulation to be deemed sufficiently consistent with the historical record. In seeking to recreate, say, 60 years of observations, an ensemble of typically 50 runs is started and run for a year. Those on track with observation are then selected and in turn copied and further perturbed to generate the restart data from which 50 runs can continue for the next year.

There are sufficient similarities in terms of the software patterns that particle filtering had been built into an early version of OptClim, and was included as a second type of workflow for this project.

Optclim overview

Figure 1 shows the cyclical workflow of OptClim.

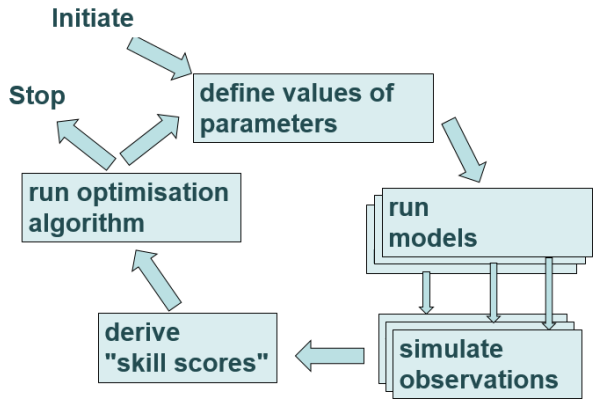


Figure 1 Overview of the OptClim workflow

1. The researcher creates an initialisation file that also gives information for each parameter and simulated observable, their covariances used to compute error; the optimisation method and criteria for ending the workflow.
2. The researcher defines starting parameter values. This causes an initial sampling of the parameter space to be made with typically P+1 parameter sets, where P is the number of parameters being optimised.
3. Typically P+1 model instances are run in parallel (described further below).
4. For each model simulated observations are derived.
5. For each a skill score is determined: the simulated observations are compared with data observed from satellites (e.g. short and long wave radiation intensities -

reflected and emitted from the Earth, respectively), in situ observations, and other estimates.

6. The optimisation algorithm is then run to define the next set of simulations, each with a new set of parameter values. The new set may be one run only or multiple runs.

The cyclical workflow continues until termination criteria are achieved such as a good agreement with observation.

The optimisation algorithm is chosen by the researcher. We currently favour using DFOLS as the optimisation algorithm: this tends to converge with fewer model runs than Gauss-Newton which we first used with OptClim.

The "run models" function entails for each instance of the model:
1. Clone an existing configuration of a model, reusing its executable(s) to avoid recompilation
2. Modify simulation parameters.
3. Execute each model (Models can run concurrently)

The terminology we use is as follows:

Table 1 Glossary of OptClim terms

Term	Meaning
Base run	A model that is cloned for each instance begun by OptClim: includes built executable(s), and has been modified by the researcher to connect with OptClim
Orchestrator	How models are managed from OptClim: - Using SLURM for CESM, MITgcm - Using Rose and GeosMeta for UKESM
PostRun script	Script run on completion of the model instance to cause the generation of the simulated observations
Run	An instance of a model known to OptClim
Scenario	Is defined by the base model - its code, input data. Example: pre-industrial, post-industrial.
Study	The full workflow executed by OptClim for a particular scenario and parameter space

The Optclim workflow for a particular study is not predetermined – results are used to identify the next model instances and parameter sets to be used. State is managed by a combination of files for the study, for each run and by dependent jobs initiated once the next set of runs is specified. The workflow required for OptClim was implemented primarily using SLURM and included short serial jobs for synchronisation, each held until completion of prerequisite jobs. This type of workflow filled the allowed serial queue limit, so led to us

requesting an increase in the per-user quota of such jobs in the queue. This was swiftly and helpfully granted by ARCHER2 systems managers and is likely to be appreciated by other workflow software in future.

Development of OptClim

The initial prototype of OptClim, termed OptClim1 was developed with funding from NERC and used on the University of Edinburgh cluster Eddie. [1]
OptClim2 was a development of OptClim1 by Prof Tett which added functionality and reimplemented some of the bash scripts of OptClim1 in Python. This has been used by Tett et al. [2] and Oliver et al, [3].

Overview of the porting to ARCHER2

Our goal was to enable use of OptClim2 with the MITgcm[4], CESM2 [5] and UKESM [6] models. Each has a cohort of users on ARCHER2 whom we hope would benefit from OptClim. We selected these due to our familiarity with them and because there are sufficient differences between the models that they stretch Optclim2 functionality.

Three phases were planned: 1) include the selected classes of models in OptClim2 with minimal change; 2) make some changes to increase the flexibility and sustainability of the software; 3) add particle filtering to OptClim2. Phase 2 proved impossible in OptClim2 to the extent that we had hoped, and so we have begun re-implementation of what will become OptClim3, with some of the new components being necessary in OptClim2 for UKESM. This is explained further below.

OptClim2 is now available for use as documented in [7]. A researcher wishing to use OptClim will need some support, initially, and we will give best-efforts support.
In phase 1 of the porting, we added support for SLURM in controlling OptClim2 workflow, and then for each model in turn. We built our work upon our previous eCSE collaborations in porting MITgcm and CESM2 to ARCHER2. [https://github.com/eCSE-MITgcm-ARCHER2/; https://docs.ARCHER2.ac.uk/research-software/mitgcm/mitgcm/]; [ref].

From the viewpoint of integrating with OptClim there is a progression in complexity through these three types of model.

1. MITgcm [https://docs.ARCHER2.ac.uk/research-software/mitgcm/]. is core software upon which a model is then built –we used ECCOv4r4 [8] as a first example, but the pattern of each MITgcm model is similar. It has a thin layer of scripts used to make the models, then sbatch is run directly by the user to submit a job on ARCHER2.
2. CESM has a comprehensive, readily used layer of scripts and workflow to create, setup, build and run a model, and to clone an existing model [9]
3. UKESM is supported for the UK academic community by the National Centre for Atmospheric Science (NCAS). A Rose/Cylc workflow suite manages, builds and runs these models on ARCHER2, from NCAS' server PUMATEST [10], with implications for OptClim that are described below.

Each model type has required functionality captured in its own python module: MITgcm.py, CESM.py, UKESM.py. These hold subclasses of the ModelSimulation superclass. They

establish new clones, set their parameters values as defined by the optimiser, queue batch jobs that generate simulated observations and rerun the optimisation algorithm. UKESM is different due to the use of Rose/Cylc on PUMATEST. The following table summarises the work needed to integrate each type of model with OptClim2.

Table 2 Summary of extensions to OptClim2 for each type of model

Function	MITgcm models	CESM2	UKESM1 (Rose/Cylc models)
Clone "base model" avoid rebuild of each clone	Copy model "run" directory (linking to, but not replicating the input data files)	Used CESM clone script	Copy a base Rose suite on PUMATEST including copying executable(s) to avoid rebuilding each clone
Amending parameters	ModelSimulation method to edit namelist(s)	CESM provides scripts that read amendments to parameters from a simple text file – so OptClim writes that text file rather than directly editing namelists.	Edit rose configuration files on ARCHER2. This is done by an additional pre-run task in the rose suite run before the model is started.
Run new model	Direct sbatch with slurm script	CESM provides script to submit a case	PUMATEST runs new rose suite.
Post-run processing task	Held job released on completion of model	CESM has its own workflow, and on completion releases the corresponding task	Include post-run task in the Rose suite.

UKESM1 and OptClim2

The NERC-Met Office's UKESM is built and run using a Rose/Cylc workflow [10]. Tasks in a Rose/Cylc suite can execute on either PUMATEST or on ARCHER2 (for the model build. Execution, processing..).

When OptClim2 needs to generate a clone of a model, instead of doing a local copy and modifying parameters (as for CESM or MITgcm) on ARCHER2 we need to call out to PUMATEST to cause it to copy/start a suite that then controls the clone. Suites must be run from PUMATEST – they cannot be run from ARCHER2 because agents corresponding to each task of a suite need to persist.

The approach we took was to have an over-arching Cylc workflow run on PUMATEST, to invoke OptClim2 on ARCHER2 and then to loop receiving requests to clone a base model and

submit the cloned suite to ARCHER2. The base model included some add-ons to link to the OptClim2 functionality.

In OptClim2 the structure is of the form:

Run optimiser
Define models' parameters
For each model
Clone
Modify
Run

When using Rose/Cylc with UKESM we need looser coupling, simply:

Run optimiser to define models' parameter sets
Send requests to PUMATEST to clone and run suites

With OptClim2 we therefore require a batch job on ARCHER2 to invoke a script on PUMATEST, to clone and run each model's suite. The two simplest approaches are not possible with PUMATEST:

- From ARCHER2, in a batch job, using ssh to run a script on PUMATEST is impossible. The security arrangement between PUMATEST and ARCHER 2 uses 2FA requiring an ssh key with passphrase. For policy reasons, ARCHER2 does not allow us to use an ssh agent to hold the key and be accessible to a batch job in the serial queue.
- Cylc "broadcast" is not supported by ARCHER2 and PUMATEST. Broadcast is the way in which a Cylc suite, run from a Cylc server, could receive data that can be used in subsequent tasks. Whether this is available depends on a combination of the policies of the involved systems, and on the installation of Cylc.

A replacement for PUMA is being planned, and this will be closer to ARCHER2 in terms of security – but details were not defined. It is hoped that the 2FA will no longer be needed and workflow across PUMA and ARCHER2 will become simpler to orchestrate.

However, wishing to progress with the project we adopted a third method, communicating between ARCHER2 and PUMATEST using a prototype service called GeosMeta [11], which we had developed here in Edinburgh. GeosMeta provides researchers with a document-oriented database in MongoDB where each document is in effect a python dictionary. Developed for use from any/all of the various computers research teams might use (PC, university cluster, ARCHER2, JASMIN…) its goal is to help in the management of provenance of research data. Users are registered and authenticated. Documents are associated with projects to which users are given rights. GeosMeta is a RESTful web-service, built on Flask [12] and EVE [13]. We tested GeosMeta on ARCHER2 and PUMATEST, using its python client software.

We poll GeosMeta from PUMATEST to receive data written to GeosMeta from ARCHER2 – establishing a message passing layer for OptClim2 between the two computers.
A Rose/Cylc workflow on PUMATEST periodically:

1. runs a task on ARCHER2 to cause it to send requests for new clones to GeosMeta. Such requests are sent after the optimisation methods has run to define the next set(s) of parameters for new suites.
2. On PUMATEST to check for new requests from ARCHER2. Such requests define the numbers of new clones, if any, that are needed at that time.

Each request made from ARCHER2 is a new document in GeosMeta with a status field that is updated when PUMATEST has satisfied the request.

OptClim2 users with UKESM need also to register with us at University of Edinburgh as a user of GeosMeta – but we hope the necessity of using GeosMeta will be lifted with the anticipated new PUMA.

Testing of OptClim2 with CESM, MITgcm and UKESM

To the existing OptClim2 test suite, we added tests for the python for each of the new models' classes.

For each of the three model types there is a simple study with a couple of exemplar parameters and simulated observables. These are documented in the GitHub pages specific to each model, at [7]. They are not scientifically meaningful, but a) test the system, and b) give users an initial case they can run to gain familiarity and then modify to define their own parameters and simulated observations. Users will need to code the generation of the simulated observations for their case. The testing used the optimisation method DFOLS [3], which is the current preferred method. At different stages of a study DFOLS requires multiple models to run or just a single new model instance to run. When multiple models are requested by DFOLS, these are submitted to SLURM to be able to run concurrently. (We did not test with Gauss Newton, envisaged in the proposal because DFOLS covered both these cases – single and multiple concurrent runs.)

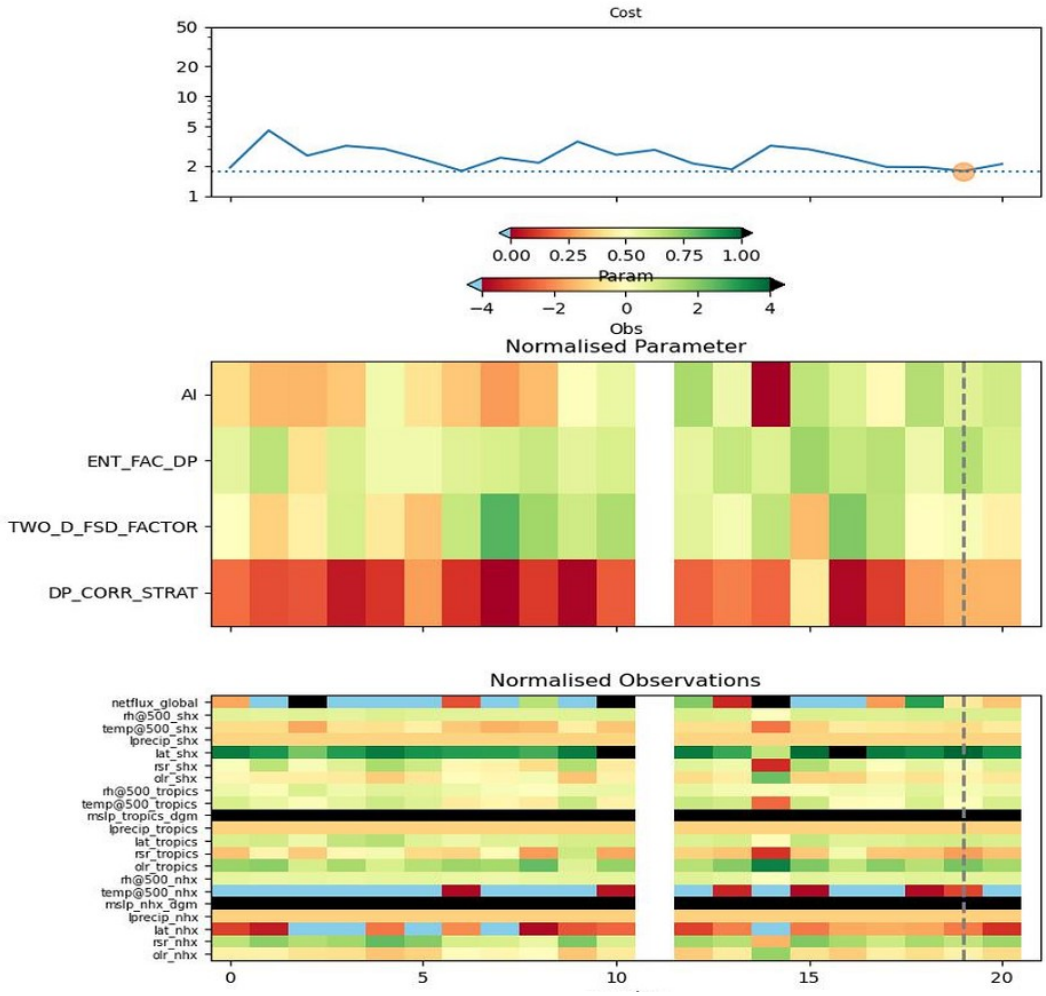
Demonstration of OptClim2 with UKESM

Using 4 parameters and 21 observations, we tested OptClim with an atmosphere only UKESM. This was a demonstration rather than a serious attempt to improve the model - that would entail judicious choice of more parameters and observations, and care with setting the optimiser's own options - but it proved that the system is capable of "real-world" use.

The base suite (that which is cloned for each run), code to generate the simulated observables, and the configuration file used to define the study are all held in GitHub [19].

The results are in Figure 2. The optimisation reduced the model-observation misfit ("cost") though only by about 20%. The algorithm modified all parameters with the final optimisation showing quite large changes.

Figure 2 Initial "real-world" trial with 4 parameters and 21 observables for an atmosphere-only UKESM. Top: cost as a function of model iteration (x-axis). Orange circle shows minimum cost. Middle – parameter values normalised by their range using "param" colour scale. 0(1) corresponds to minimum (maximum) allowed value. Y-axis is parameter, x axis model iteration. Bottom observed value normalised by estimated error (largely observational error). X-axis is model iteration and y-axis is observation. Values below (above) -4 (+4) are show in light blue (black) respectively. The colour scale is show on the "obs" colour scale. vertical dashed line in middle and bottom plot shows minimum cost value.



Particle Filtering (PF) case

In this case an ensemble of perturbed models is run for a short period, then filtered by comparing against a historical record. The runs most aligned are used with small perturbations as initial conditions for the next period, with runs inconsistent with the past data stopped . The perturbations are applied to the "dump files" which are used by climate models to hold state and allow restarts.

Our initial intention was to use OptClim2 code, replacing the OptClim2 optimisation algorithm by the reorganisation of the dump files and then starting new runs. We had taken this approach in an earlier implementation on our university cluster, with a different model, using OptClim1. However we decided to use some patterns of software from OptClim2 on ARCHER2, but have a different code-base for the PF. The contrasts with the first implementation of the PF include: a) the cloning of a suite is trivial compared to the earlier

model that required extensive edits of files; b) we have experience and could reimagine how PF should work (to set up one ensemble and pause and restart it, rather than a large number of short new runs, then needing to be post processed to connect them); c) by using a overriding PF suite (to invoke and then manage the restart of the model suites) we had relatively little added code to write for the PF core system itself; and d) whereas OptClim manages its workflow with dependencies between jobs, the number of jobs with the particle filtering could easily become excessive, so a different approach was used, polling to recognise when previous jobs had all completed and the filtering should be run

This PF suite uses GeosMeta in similar ways to those of OptClim2 – to send messages from ARHCER2 to PUMATEST, in this case to cause a run to be restarted. It clones a base suite in similar ways to OptClim with pre-run and post-run tasks inserted in the Rose/Cyllc suite..

The PF outline is as follows.
From PUMATEST start a particle filtering suite that:

- Starts the required number of model suites, to run a model for one time interval (e.g. 1 year). Each model suite has pre and post tasks inserted in the workflow.
    - On first invocation a "reuse" task copies already built executables and drivers
    - A pre-model task sets the state of the run to RUNNING, by making a flag file associated with the model instance.
    - The post-model task generates a goodness of fit, comparing the current state to the historical record and resets the flag file to state DONE.
- Periodically
    - Asks ARCHER2 to check the status of model runs. When all runs are DONE
        - the filtering is run to reset the dumps of the ensemble members
        - Messages are written to GeosMeta
    - On PUMATEST , checks GeosMeta for messages that request a restart for models, updates the end-of-run time to add the required length of this run, and restarts each model

The goodness of fit, and the algorithm for filtering dumps is specific to the researcher's goals, so has to be developed to a straightforward interface provided by the PF software[14]

Towards OptClim3

We were seeking a more flexible, configurable, easily deployed and more sustainable version of the software to be termed "OptClim3." We envisaged modules that could be readily linked via the SLURM jobs as in OptClim2 for models such as MITgcm and CESM, or run from a Cylc suite for models such as UKESM. We also wished to simplify the addition and modification of more complex optimisation methods, and the associated covariant analyses that are applied to the simulated observations.

This entailed a) improving the modularity – to more clearly reflect the basic OptClim functions of define runs, clone, modify, run, postprocess runs (using covariance and optimisation methods) and loop again; b) using a database to hold parameter sets, runs' status and simulated observables. The database permits more flexible workflow in future, rather than waiting for all runs to finish before reassessing if more can be specified. The

database also enables improved recognition of when a parameter set has already been run so results can be reused rather than recreated.

For OptClim3 we now have:
1. a database that contains the state for multiple studies defined by a set of parameters each consisting of multiple scenarios (A scenario might be pre-industrial and post-industrial era).
2. a web app that manages access so that the database can be used safely in parallel; and a system that can scale.
3. web requests come at a cost, i.e. they are an order of magnitude slower than accessing a local database. This is particularly problematic as the optimiser queries the system for results of previous runs. Lookups have to be repeatedly made for all runs in the previous iteration thus processing time for each iteration is increasing with the number of iterations. The test setup took days for 100 iterations when the database is queried via the web application. With caching the same test run completes within 1 hour.
4. since it is only the optimiser that does this form of access we can cache the particular lookup locally. In this case we don't need to worry about parallel access or the number of parameters. The cache database is a single table.
5. this system is completely independent of the orchestrator (e.g. software running the optimisation and determining next runs), and middleware used for scheduling. The only requirement is that all workers (including the optimiser) see the same filesystem.
6.The database holds the state of the system. The optimiser queries the database for existing runs and adds parameters of new runs. The workers retrieve a new parameter set from the database and record state transitions.
7. we have a command line manager that can do some basic querying of the state of a study/scenario.
8. in future we can add a web frontend to the system

The OptClim3 system consists of a number of python packages:

1. The **ObjectiveFunction-server** module is a flask web application that provides a REST API to interact with the objective function lookup cache. It use a relational database such as postgresql to store the cache. Typically a webserver such as apache2 will be put in front of the flask application. [15]
2. The **ObjectiveFunction-client** module communicates with the server via HTTP REST calls. The module maintains a local cache of parameter sets in the COMPLETED state. It comes with a utility, **objfun-manage**, that can be used to manage entries in the lookup table. [16]
3. The **ModelOptimisation2** module uses the client module and is specifically designed for optimisaing climate models. This module contains a cylc8 workflow that controls an optimisation run. [17]

A simple Fortran program [18] that uses namelists for configuration and produces a netCDF output file is also available to demonstrate the system in place of a real climate model.

The basic OptClim3 system is complete. Database and web application associated with OptClim3 are deployed on School of GeoSciences servers. Additional funding will be required to adapt one of the climate models.

Summary and Status

OptClim2 and Particle Filtering are currently being set up for initial research use by the developing group, with UKESM. We are planning to use the ARCHER2 webinar series of seminars to inform the CESM, MITgcm and UKESM user communities of the availability of OptClim2 and PF. We will continue to use the GitHub repositories to communicate progress with any code enhancement, and with documentation for users. With best efforts we will support others wishing to trial the software.

We also wish to see the new PUMA deployed as this will be more closely linked to ARCHER2 and is likely to allow us to reduce the complexity of both OptClim2 and PF with UKESM. That will be the time to build on our experience with the software, potentially simplify it due to the new PUMA characteristics, and invite wider use.

References

1. Tett, S. F. B., Yamazaki, K., Mineter, M. J., Cartis, C., and Eizenberg, N.: Calibrating climate models using inverse methods: case studies with HadAM3, HadAM3P and HadCM3, Geosci. Model Dev., 10, 3567–3589, https://doi.org/10.5194/gmd-10-3567-2017, 2017
2. Tett, S., Gregory, J. M., Freychet, N., Cartis, C., Mineter, M., & Roberts, L. (2022). Does Model Calibration Reduce Uncertainty in Climate Projections? Journal of Climate, 2585–2602. https://doi.org/10.1175/JCLI-D-21-0434.1
3. Oliver, S., Cartis, C., Kriest, I., Tett, S. F. B., and Khatiwala, S.: A derivative-free optimisation method for global ocean biogeochemical models, Geosci. Model Dev., 15, 3537–3554, https://doi.org/10.5194/gmd-15-3537-2022, 2022.
4. http://mitgcm.org/
5. https://www.cesm.ucar.edu/models/cesm2/
6. https://ukesm.ac.uk/
7. https://github.com/optclim/ModelOptimisation/wiki
8. https://ecco-group.org/products-ECCO-V4r4.htm
9. https://esmci.github.io/cime/versions/master/html/users_guide/cloning-a-case.html
10. https://cms.ncas.ac.uk/rose-cylc/
11. https://www.wiki.ed.ac.uk/display/GeosMeta/GeosMeta
12. https://flask.palletsprojects.com/en/2.1.x/
13. https://docs.python-eve.org/en/stable/
14. https://github.com/Climate-Particle-Filtering
15. https://github.com/optclim/ObjectiveFunction-server
16. https://github.com/optclim/ObjectiveFunction-client
17. https://github.com/optclim/ModelOptimisation2
18. https://github.com/optclim/DummyModel
19. https://github.com/SimonTett/OptClim_UKESM