

# User Guide: Applying Optimisation Algorithms to Climate Models with OptClimSO

Sophy Oliver Department of Earth Sciences, University of Oxford [sophy.oliver@oriel.ox.ac.uk](mailto:sophy.oliver@oriel.ox.ac.uk)

June 25, 2021

## 1 Introduction

OptClim is a Python-based control system which allows any computational model to be optimised (or "tuned") by any optimisation algorithm. Neither the climate model or the optimisation algorithm need know much about the other, only their inputs and outputs, which the OptClim system reads, monitors and distributes. This way, the climate model, which are usually very computationally expensive and complex, can be treated as a "black box".

## 2 Inheritance and Acknowledgements

OptClim (<https://github.com/SimonTett/ModelOptimisation>) was created and written by Professor Simon Tett, University of Edinburgh, and used in the optimisation studies Tett et al. (2013) and Tett et al. (2017). In 2018 OptClimV2 was edited by Sophy Oliver, University of Oxford, to include the optimiser CMA-ES, into version OptClimSO.

## 3 Source

Version described in this user guide: <https://github.com/SophyOliver/OptClim>.  
Original source: <https://github.com/SimonTett/ModelOptimisation>

## 4 Installation

Copy all of OptClim (<https://github.com/SophyOliver/OptClim>) across to your working area. This currently requires Python 2 with all necessary packages (e.g. pandas, numpy, scipy). See [https://github.com/SophyOliver/OptClim/blob/master/Load\\_Optclim\\_Packages.txt](https://github.com/SophyOliver/OptClim/blob/master/Load_Optclim_Packages.txt) on how to load these.

To compile the optimisation algorithm package you would like to use, do the following.

1. Create subfolder (e.g DFOLS, or Py\_BOBYQA, or cmaes).

2. Copy in required code. Py\_BOBYQA can be found here:

<https://github.com/numericalalgorithmsgroup/pybobyqa>, DFOLS here:

<https://github.com/numericalalgorithmsgroup/dfols>, and the CMA-ES optimisation framework can be found in the Supplement of Kriest et al. (2017), though the script cmaes.cpp has edited to make it compatible with OptClim and is here:

<https://github.com/SophyOliver/OptClim/blob/master/cmaes/cmaes.cpp>.

3. Within DFOLS and Py\_BOBYQA subfolders type "pip install ." to install the algorithms, and within the cmaes subfolder, type "make" into the command line to compile.

## 5 Workflow

OptClim allows a climate model to be run multiple times to iteratively improve the model parameters. It does this by using a provided optimisation algorithm to iteratively minimise the model error (or "misfit"). As shown in the workflow diagram in Figure 1, the continuous loop of running a model then an optimiser is controlled by a JSON file in OptClim. The climate model in this example requires the input file "parameters\_input.txt" and is run by the script "runscript". This model then runs, internally using any observational files, initial conditions files or scripts it needs to, and calculates it's misfit error to be minimised. The optimiser then reads in the previous history of parameter values and associated misfits, and determines the next parameter set to try which is likely to minimise the misfit. The model is then run again with this parameter set, and the loop continues until the optimiser determines that the misfit has been sufficiently minimised.

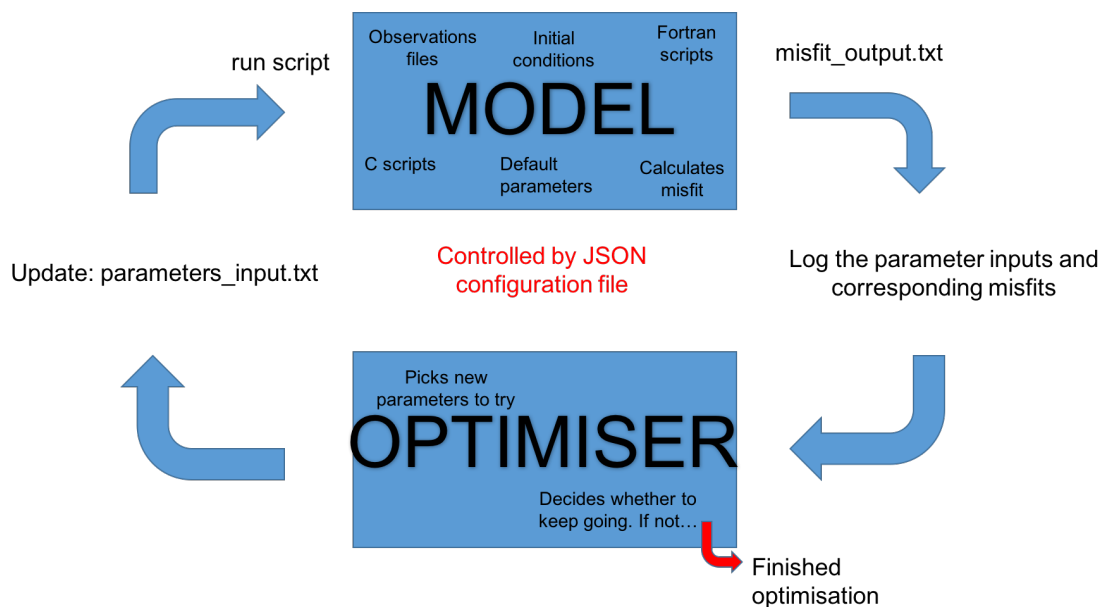


Figure 1: Schematic of OptClim's workflow.

## 6 OptClim Scripts

### 6.1 JSON File

The JSON (JavaScript Object Notation) file is written in a way that is easy for machines to parse, as well as for us to read and write. Here the JSON file contains all the information provided by the user regarding their optimisation study, model parameters, location of reference files and scripts, optimiser to use, optimisation settings, etc. For examples see folder OptClim/Configurations/OxfordMOPS\_Configs/TWIN\_Configs.

### 6.2 runOptimise.py

runOptimise.py is a Python script which begins the workflow loop shown in Figure 1. Inputs it requires include an output directory to save each iteration's climate model results to, and the JSON file specifying the OptClim settings. Below is an example of how to call runOptimise, with “-restart” specifying it to delete any previous iterations and start the optimisation study from scratch, and “&>progress.txt” specifying all text output to the command line to be printed in a text file.

```
python runOptimise.py - -restart -dir outputDirectory -v -t
$JSON_FILE_LOCATION_myJsonFile.json & >outputTest/progress.txt
```

#### CODE DESCRIPTION: RUNOPTIMISE.PY

INPUT: Output directory (“rootDir”), JSON file location, and if it is to RESTART.

1. Import packages.
2. Define functions.
3. Define the function for model creation using ModelSimulation.py and the class specific to the climate model.
4. Use StudyConfig.py to load in the config information from the JSON file.
5. Create MODELRUN to store all information on models previously run.
6. **if** RESTART is True **do**
7.     Delete all files and folders in the output directory.
8. **end if**
9. **for** every subfolder in rootDir **do**
10.     Read in that model run's information and store in MODELRUN.
11. **end do**
12. Read algorithm name from config.
13. **if** algorithm name == “algorithmX” **do**
14.     Retrieve user's settings specific to this optimisation algorithm.
15.     Run the optimiser.
16.     **if** optimiser states it's not yet at a solution **do**
17.         Append MODELRUN[‘modelsToMake’] with a new model run(s).
18.         Raise “LinAlgError”: Go to line 22.
19.     **end if**

```
20. end if
21. Print and save solution status.
22. if LinAlgError do
23.     for every model in MODELRUN['modelsToMake'] do
24.         Create the next subfolder in rootDir.
25.         Create the model for this subfolder using ModelSimulation.py and the class
            specific to the climate model.
26.         Submit the model run using Submit.py.
27.     end do
28. end if
29. Store optimisation history and updated configuration file.
30. Write optimisation history to the "final" JSON file.
```

## 7 Two job submission workflow options

### 7.1 Option 1: Individual Sequential Job Submissions

If your model runtime is large, or maximum cluster wall clock time is short, then you will need to submit each iteration of the optimisation as an individual job. You do this using the python script `runOptimise.py`, and start the optimisation via the following:

```
# Load python and packages (see
# https://github.com/SophyOliver/OptClim/blob/master/Load_Optclim_Packages.txt).
module load python/anaconda2/5.0.1
source activate $DATA/myenv

# Call runOptimise.py
python runOptimise.py - -restart -dir out_folder -v $JSON_Location/myJsonFile.json
```

### 7.2 Option 2: One Single Large Job Submission

If your model runtime is small, or maximum cluster wall clock time is large, then you can avoid entering the submission queue before each optimisation iteration by submitting the entire optimisation as one large single job, thereby only waiting in the submission queue once. You do this using the python script `runOptimise_oneJob.py`, and start the optimisation via the bash script `optscript` (for example: <https://github.com/SophyOliver/OptClim/blob/master/optscript.sh>). Note that, for this option, job submission commands (regarding nodes, wall clock time, etc.) should no longer be in your runscript, but in your `optscript`.

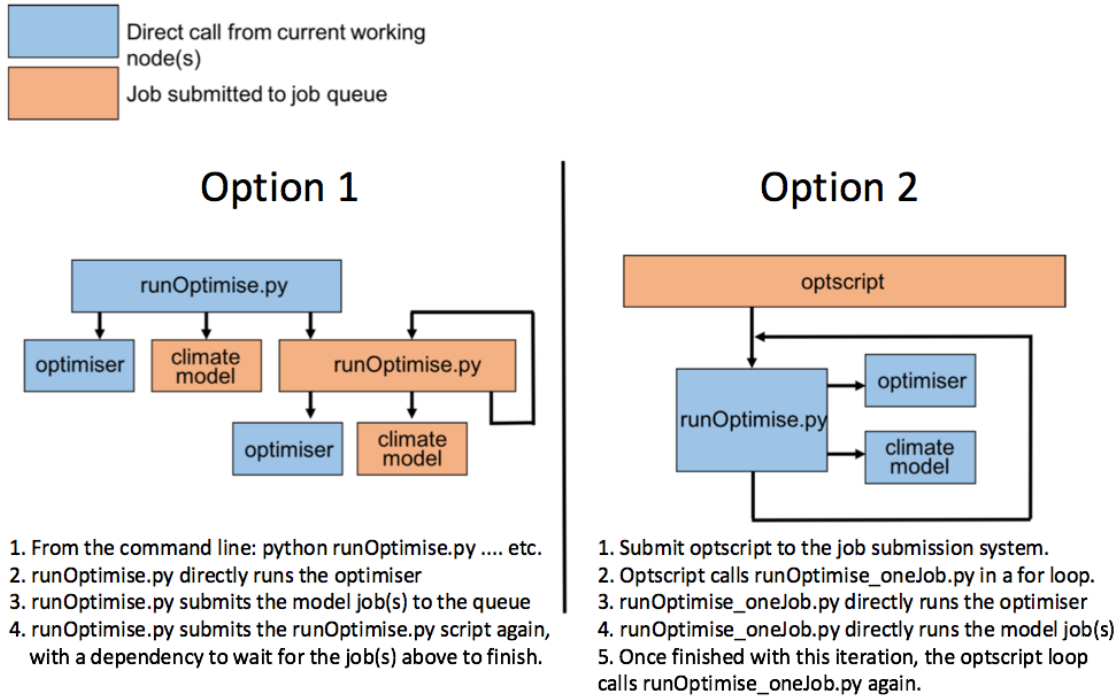


Figure 2: Schematic of 2 job submission workflow options.

## 8 Using OptClim

### 8.1 Step 1: Setting up

1. In .bash\_profile or .bashrc make sure \$OPTCLIMTOP is location to your local copy of the OptClim folder, and that OPTCLIMTOP is in your PYTHONPATH.
2. Create conda environment (see [https://github.com/SophyOliver/OptClim/blob/master/Load\\_Optclim\\_Packages.txt](https://github.com/SophyOliver/OptClim/blob/master/Load_Optclim_Packages.txt))

### 8.2 Step 2: Adding a new climate model

Inputs (e.g. parameter values) and outputs (e.g. misfit error to minimise) specific to the climate model need to be sorted by a Python class. For example see <https://github.com/SophyOliver/OptClim/blob/master/OxfordMOPS/MOPS.py>. This class should use ModelSimulation.py to inherit all necessary files from the rootDir (which at least contains your runscrip) and place a copy in the new model folder(s). To introduce a new climate model to OptClim, see the following instructions.

1. Under \$OPTCLIMTOP/Configurations create a new \_Configs folder named appropriately for the model e.g. OxfordMOPS.Configs.
2. Within your new \_Configs folder place all the RunCode (e.g. runscrip). This is the ROOTDIR, and will get replicated for every algorithm iteration.
3. Create a new folder under OptClim labelled according to the model you're using

- (e.g. OxfordMOPS) and add it to your PYTHONPATH in .bash\_profile.
4. In this folder copy the contents of <https://github.com/SophyOliver/OptClim/tree/master/OxfordMOPS>.
  5. Create a python script creating the model class. An example of this is: <https://github.com/SophyOliver/OptClim/blob/master/OxfordMOPS/MOPS.py>. If your runscript reads “parameters.input.txt” and writes “misfit\_output.txt” then this example script MOPS.py can be used almost exactly as it is, just change the script name and any mention of “MOPS” to your chosen model class name.
  6. Edit runOptimise.py and/or runOptimise\_oneJob.py so that they call your model class. (e.g. replace MOPS.MOPS with your class name).
  7. Edit Submit.py to correctly submit to your submission system by adding a new definition (**def**) and calling this from runOptimise.py (line 698).

### 8.3 Step 3: Adding a new optimisation algorithm

There are currently 4 optimisation algorithms incorporated into OptClim. Py\_BOBYQA (Cartis et al, 2018) is a Python-based implementation of BOBYQA (Bound Optimisation BY Quadratic Approximation), which is an iterative method of minimising a function  $f(x)$  with bounds on the variables, by Powell (2009). The Python implementation of derivative-free optimisation using least squares (DFOLS) algorithm (Cartis et al, 2018) works very similar to Py-BOBYQA, but takes more of the problem structure into account. The most recent version of Py\_BOBYQA can be found here:

<https://github.com/numericalalgorithmsgroup/pybobyqa>. The most recent version of DFOLS can be found here: <https://github.com/numericalalgorithmsgroup/dfols>. Both Py\_Bobyqa and DFO-LS are derivative-free optimisation algorithms, while Gauss-Newton requires derivatives. These 3 optimisers were incorporated into OptClim by Prof. Simon Tett, and are deterministic local optimisers. A stochastic global optimisation method requiring more function evaluations is the Covariance matrix adaptation evolution strategy (CMA-ES) (Hansen, 2016), which is nicely explained in application by Kriest et al (2017). To introduce a new optimiser to OptClim, see the following instructions.

1. Download and compile the new optimiser to it’s own sub directory within OptClim and add the necessary paths.
2. Edit runOptimise.py and/or runOptimise\_oneJob.py so that they load your chosen optimiser.
3. Code another branch to the **if** statement in the main block of code of runOptimise.py, to be executed in the case of the user specifying this new optimiser. This branch should collect the inputs required for the optimiser, run the optimiser, then read the information output by the optimiser needed for OptClim to continue.

### 8.4 Step 4: Create your JSON file

1. Create a JSON configuration file specific to this model and optimisation. Don’t have this in ROOTDIR. This file should contain the initial conditions of the parameters, their ranges, etc. (see examples within <https://github.com/SophyOliver/OptClim/tree/master/Configurations>

/OxfordMOPS\_Configs/TWIN\_Configs).

## 8.5 Step 5: Test all works

1. Test the first iteration at the command line using “restart”:

```
python runOptimise.py - -restart -dir outputTest -v -t
$JSON_FILE_LOCATION_myJsonFile.json & >outputTest/progress.txt
```

- “t” specifies to run only one iteration.
- This runs the first optimisation iteration by copying everything from ROOTDIR into a subfolder within the output folder called outputTest.
- The restart command clears every test folder within outputTest and starts from the beginning.
- Check all works and the output test folder contains the correct model outputs, parameter\_inputs.txt and misfit\_outputs.txt.
- If getting errors do `python -m pdb runOptimise.py ...` etc. then put in breakpoints (b 354), `r=run`, `c=continue`, and `n=next`, to find the problem.
- The `& >.txt` at the end outputs all the terminal statements to a text file

2. Test second iteration at the command line:

```
python runOptimise.py -dir outputTest -v -t
$JSON_FILE_LOCATION_myJsonFile.json & >outputTest/progress.txt
```

- This carries on from whichever iteration it got to in this outputTest folder
- Note “- - restart” is not used here, which would delete all previous folders and begin the optimisation from the beginning.

## 8.6 Step 6: Set off an optimisation

If using `runOptimise.py` (workflow option 1), you set off an optimisation by typing this into the command line, after activating your conda environment:

```
python runOptimise.py - -restart -dir outputDirectory -v -t
$JSON_FILE_LOCATION_myJsonFile.json & >outputTest/progress.txt
```

Usually the maximum number of evaluations, set in the JSON file, will determine when to terminate the optimisation. Once completed, you can run the above command line again (no further models will be submitted as it knows it has completed), and `outputTest/progress.txt` will then contain the updated information for the whole optimisation.

If using `runOptimise_oneJob.py` (workflow option 2), you set off an optimisation by submitting your optscript to the cluster job queue, with enough wall clock time allowed for your specified number of optscript loops to be completed.

At the beginning of each iteration, the ...final.json file in the outputDirectory folder will be updated to include the previous iteration's parameter and misfit information. This information can then be used for plotting results.

## 9 OptClimSO Plotting Scripts

Source:

[https://github.com/SophyOliver/OptClim/tree/master/OPTCLIMSO\\_PlottingScripts](https://github.com/SophyOliver/OptClim/tree/master/OPTCLIMSO_PlottingScripts)

Here are several plotting MATLAB scripts to visualise the progress of optimisation. Some edits will need to be made for different cases.

<b>Top Script</b> — <b>Function Called</b>	<b>Purpose</b>
<b>plotAll.m</b>	Plot the progress of OptClim in this output directory, if using BOBYQA or DFO-LS.
— checkOptMatches.m	Check that the model run inputs/outputs match the text printed by BOBYQA / DFO-LS.
— plotParamTraject.m	Plot the parameter values throughout the optimisation by BOBYQA / DFO-LS. See example in Figure 3.
— plotTracerMisfit.m	Plot the total and individual misfit values throughout the optimisation by BOBYQA / DFO-LS. See example in Figure 4.
— plotParamPairMisfits.m	In one figure plot all parameter and total misfit values throughout the optimisation by BOBYQA / DFO-LS for each parameter pair. See example in Figure 5.
— plotParamPairProgress.m	In one figure per iteration, plot the parameter and total misfit values during the optimisation by BOBYQA / DFO-LS for each parameter pair. See example of a single iteration in Figure 5.
<b>plotAllCmaes.m</b>	Plot the progress of OptClim in this output directory, if using CMA-ES.
— plotParamTrajectCmaes.m	Plot the parameter values throughout the optimisation by CMA-ES.
— plotTracerMisfitCmaes.m	Plot the total and individual misfit values throughout the optimisation by CMA-ES.
— plotParamPairMisfitsCmaes.m	In one figure plot all parameter and total misfit values throughout the optimisation by CMA-ES for each parameter pair.
<b>plotAllCompare.m</b>	Compare the progress of OptClim by different optimisers in these output directories.



— plotCompare.m	Create 2 plots. (1) Plot on top of each other the total misfit values throughout the optimisation by different optimisers. See example in Figure 6. (2) Plot on top of each other the parameter values throughout the optimisation by different optimisers. See example in Figure 7.
— plotCompareMisfits.m	Compare the individual misfit values throughout the optimisation by different optimisers. See example in Figure 8.
— plotParamRelations.m	Create one figure per parameter pair comparing the parameter tuning of one parameter pair by 2 different optimisers. This helps spot any non-independent parameters which may influence the tuning process. See example in Figure 9.

Table 1: OptClimSO Plotting Scripts.

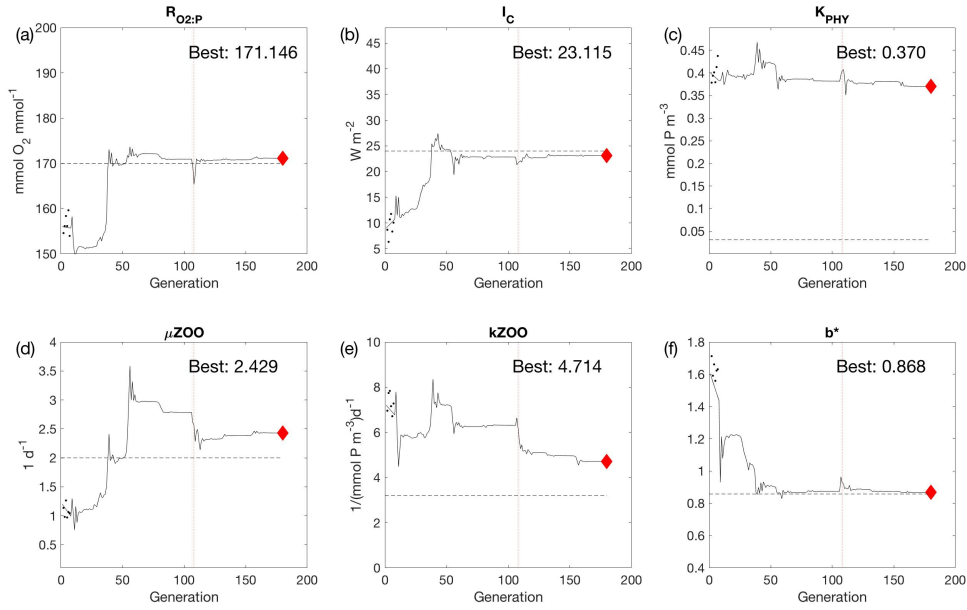


Figure 3: Tuning 6 parameters by DFOLS, created by plotParamTraject.m.

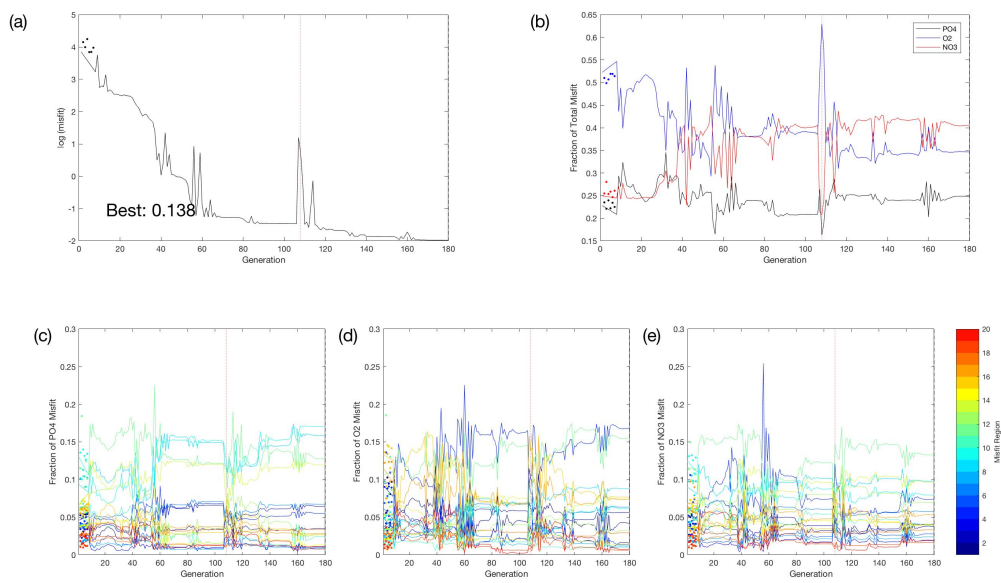


Figure 4: Minimising the misfit with DFOLS, created by plotTracerMisfit.m. Misfits have been broken down into different tracers/regions, which is very specific to this case.

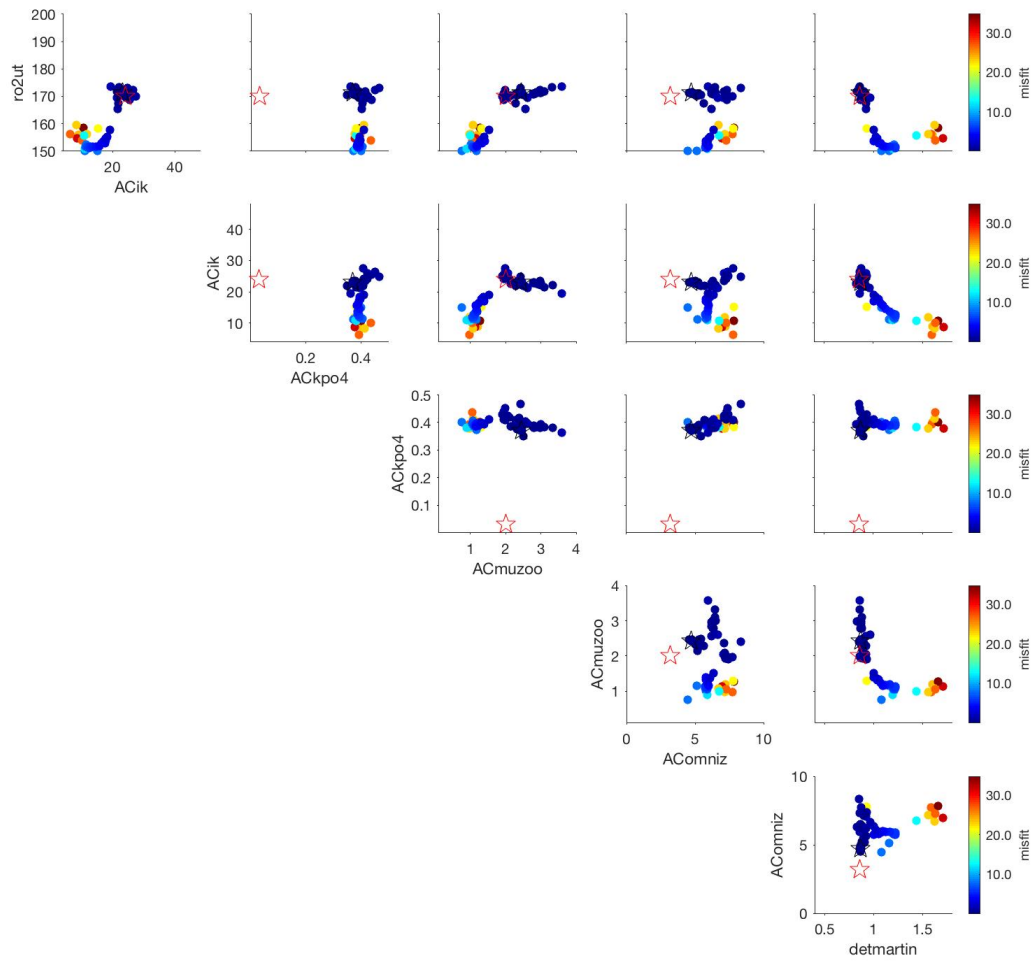


Figure 5: How the misfit is reduced in relation to each parameter pair being tuned by DFOLS, created by `plotParamPairMisfits.m`.

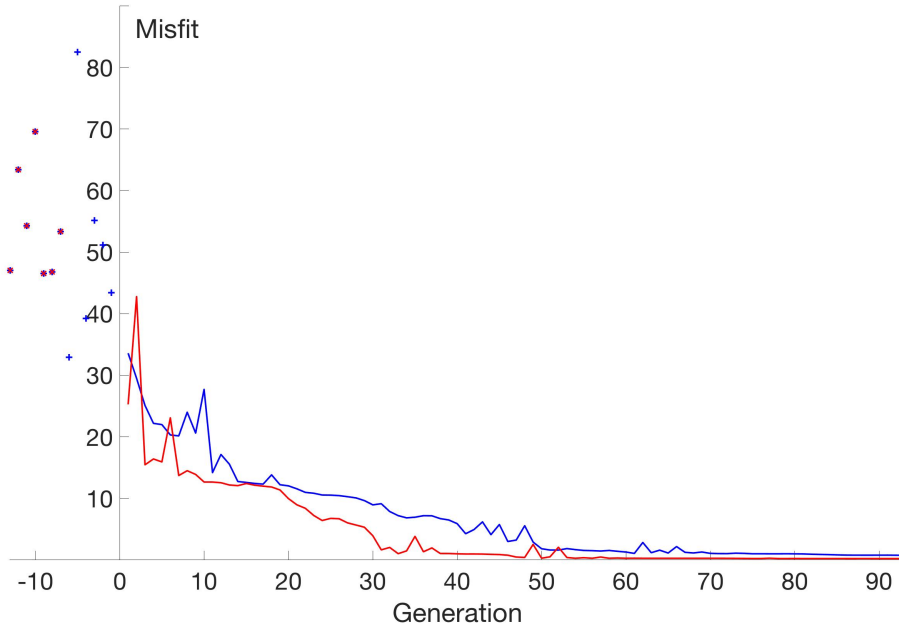


Figure 6: Minimising the misfit with DFOLS and BOBYQA, created by plotCompare.m.

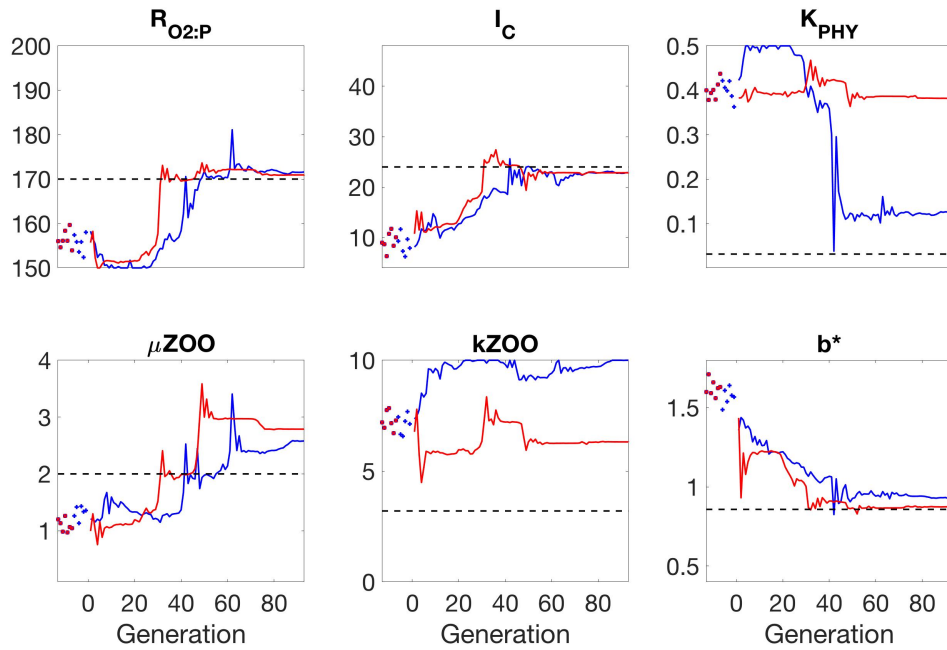


Figure 7: Tuning 6 parameters by DFOLS and BOBYQA, created by plotCompare.m (similar to plot created by plotParamTraject.m).

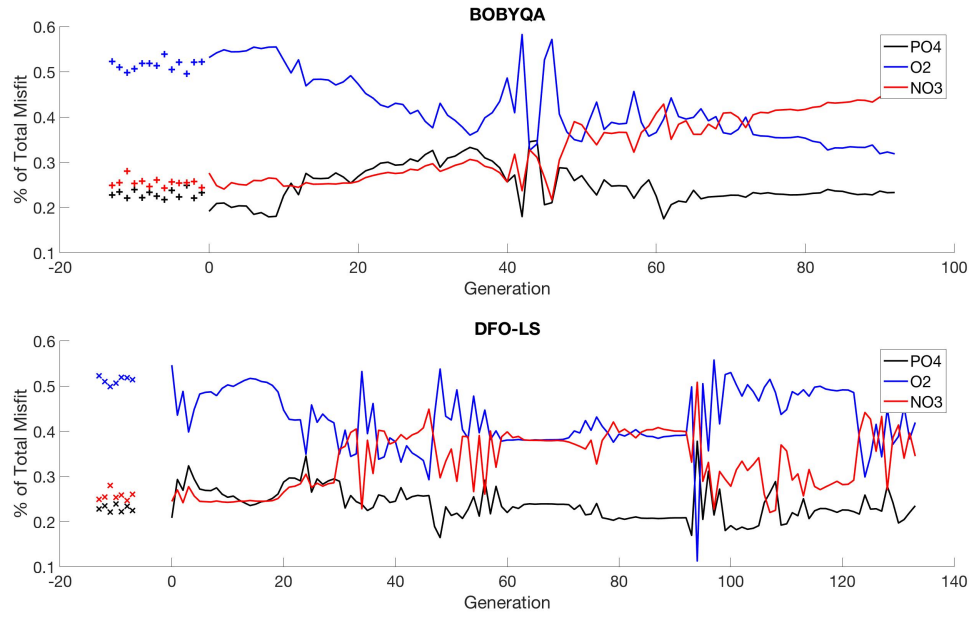


Figure 8: Minimising 3 individual misfits with DFOLS and BOBYQA, created by plotCompareMisfits.m. Misfits have been broken down into different tracers, which is very specific to this case.

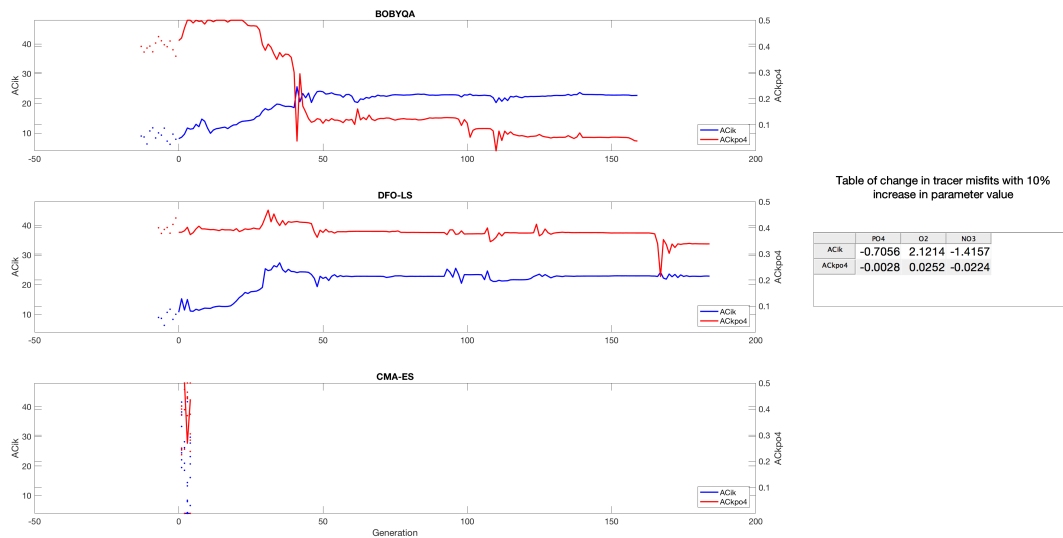


Figure 9: Compare the tuning of each parameter pair by multiple different optimisation algorithms, created by plotParamRelations.m.

## 10 References

- Cartis, C., Fiala, J., Marteau, B. & Roberts, L. (2018), ‘Improving the Flexibility and Robustness of Model- Based Derivative-Free Optimization Solvers’, arXiv preprint arXiv:1804.00154 pp. 1-35. URL: <http://arxiv.org/abs/1804.00154>
- Hansen, N. (2016), ‘The CMA Evolution Strategy: A Tutorial’, arXiv preprint arXiv:1604.00772 . URL: <http://arxiv.org/abs/1604.00772>
- Kriest, I., Sauerland, V., Khatiwala, S., Srivastav, A. & Oschlies, A. (2017), ‘Calibrating a global three- dimensional biogeochemical ocean model (MOPS-1.0)’, *Geoscientific Model Development* 10(1), 127-154.
- Powell, M. (2009), ‘The BOBYQA algorithm for bound constrained optimization without derivatives’, NA Report NA2009/06 p. 39. URL: <http://www6.cityu.edu.hk/rcms/publications/preprint26.pdf>
- Tett, S. F. B., Rowlands, D. J., Mineter, M. J. & Cartis, C. (2013) Can top-of-atmosphere radiation measurements constrain climate predictions? Part I: Tuning. *J. Clim.* 26, 9367-9383.
- Tett, S. F. B., Yamazaki, K., Mineter, M. J., Cartis, C., & Eizenberg, N. (2017): 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>.