SUMMER RESEARCH INTERNSHIP
IMPERIAL COLLEGE LONDON

on

Systematic benchmarking of different optimisation methods for mathematical modelling

Project Progress Report II

Imperial College London

Project Supervisor
Dr. Reiko Tanaka
Faculty of Engineering, Department of Bioengineering
Imperial College London
(Guided by PhD Tara Ahmed Hameed)

Submitted By

Harshita Ojha,
4th Year Integrated BS+MS
Biological Sciences
Department of Biotechnology
IIT Madras

Ayush Verma,
2nd Year Undergraduate
Bachelors in Technology
Biochemical Engineering and Biotechnology
IIT Delhi

Contents

1. PyPESTO Implementation

1.1 Encountering errors(list of errors, approach to debug it)

Post creating the extension of yaml files with details of lowerbound and upperbound for parameters etc, PEtab files were generated through pypesto.petab.PetabImporter.from_yaml(). The example below demonstrates how the function is called and assigned to a variable named “importer”. Running this command, AttributeError “tuple object has no attribute handle” was encountered as we see here.

```
import pypesto
import pypesto.petab
import pypesto.optimize as optimize
import pypesto.visualize as visualize

# import PEtab problem
importer = pypesto.petab.PetabImporter.from_yaml(os.path.join(PEtab_dir, PEtab_yaml_name),
                                                 model_name=model_name)
problem = importer.create_problem()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-63f1956a1cc6> in <module>()
      6 # import PEtab problem
      7 importer = pypesto.petab.PetabImporter.from_yaml(os.path.join(PEtab_dir, PEtab_yaml_name),
----> 8                                                  model_name=model_name)
      9 problem = importer.create_problem()

                                  2 frames
/usr/local/lib/python3.7/dist-packages/petab/yaml.py in load_yaml(yaml_config)
    125         return yaml_config
    126
--> 127     handle = get_handle(yaml_config, mode='r').handle
    128     return yaml.safe_load(handle)
    129

AttributeError: 'tuple' object has no attribute 'handle'
```

Benchmark models were tried to check whether it is yaml file extension(to generate petab files) or pypesto part that is throwing error. After playing with the Boehm Benchmark Model we got to know that there was something wrong with pypesto as extended yaml input files used from the Boehm Benchmark Model problems threw the same error.

Pypesto was tried on the Luracelli_CellSystems Benchmark Model, but the same error remained.

Overall, two approaches were tried in the above implementation (declaring importer and problem variable in Boehm Model), that is, first importing the file (initializing Importer variable-this is where we get the error, and then creating Problem variable) and the other was first setting the path and then converting it into Problem. Even with a changed approach, from <using petab importer and then creating problem> to <directly importing and creating problem> with an input of .yaml file, the same error occurred.

```
#model_name = "Zheng_PNAS2012"
model_name = "Boehm_JProteomeRes2014"
#model_name = "Fujita_SciSignal2010"
#model_name = "Sneyd_PNAS2002"
#model_name = "Borghans_BiophysChem1997"
#model_name = "Elowitz_Nature2000"
#model_name = "Crauste_CellSystems2017"
#model_name = "Lucarelli_CellSystems2018"
#model_name = "Schwen_PONE2014"
#model_name = "Blasi_CellSystems2016"

# the yaml configuration file links to all needed files
yaml_config = os.path.join(folder_base, model_name, model_name + '.yaml')

# create a petab problem
petab_problem = petab.Problem.from_yaml(yaml_config)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-24-4987be440d28> in <module>()
     16
     17 # create a petab problem
---> 18 petab_problem = petab.Problem.from_yaml(yaml_config)

                              1 frames
/usr/local/lib/python3.7/dist-packages/petab/yaml.py in load_yaml(yaml_config)
    125         return yaml_config
    126
--> 127     handle = get_handle(yaml_config, mode='r').handle
    128     return yaml.safe_load(handle)
    129

AttributeError: 'tuple' object has no attribute 'handle'
```

This link contains discussion of the error that we encountered above with yaml2sbml developer Dr. Jakob V:
https://docs.google.com/document/d/1pV9hxJI84V7Q1YruiEM3WbesBZ1n2bhFWRniyNdds1U/edit

The errors encountered above are not generated due to errors in file format or any user side glitch. Hence it can not be debugged by changing or running fresh files as it's tested above(checked two fresh models but got the same error). It is an internal error that could be handled with the correct updated package and their installed dependencies. To confirm this statement, more trials that are discussed below were done.

- Assuming path of different packages could be an issue:

Checked the path of the directories where the packages are installed with where these packages are being imported(maintained virtual environment in few cases for specific package dependencies) - threw exact error, shows path of directories wasn't any issue. Most of the techniques to debug the error eg. upgrading and downgrading different types of libraries, checking dependencies and installing all the dependencies used, but still the problem persisted.

- Assuming version of amici package could be an issue:

With amici version = 0.10.21, encountered with FileNotFoundError file not found error : Model.ODE_(something).cpp error.
Using the latest amici version = 0.11.17 and pypesto version = 0.0.13, notebook crashed when pypesto.amiciobjective() function was called.

Note: amici and pypesto were installed in correct env i.e. python3.7/dist-packages env

(**ideally with latest updated packages, one should not be facing these error issues, and should solve the issue as suggested by yaml2sbml developer**)

- Assuming google colab could be an issue:

Then to check whether there was something wrong with colab, we checked pypesto on kaggle, but found kaggle to be giving installation error in using amici (module not found error):
amici.plotting error (**module not found error**)

```
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-12-a9cb4d5d8e16> in <module>
      3 import pypesto.optimize as optimize
      4 import pypesto.visualize as visualize
----> 5 import amici
      6 import petab
      7

/opt/conda/lib/python3.7/site-packages/amici/__init__.py in <module>
    118 if not _imported_from_setup():
    119     if has_clibs:
--> 120         from . import amici
    121         from .amici import *
    122

/opt/conda/lib/python3.7/site-packages/amici/amici.py in <module>
     33         except ImportError:
     34             return importlib.import_module('_amici')
---> 35     _amici = swig_import_helper()
     36     del swig_import_helper
     37 elif _swig_python_version_info >= (2, 6, 0):

/opt/conda/lib/python3.7/site-packages/amici/amici.py in swig_import_helper()
     32             return importlib.import_module(mname)
     33         except ImportError:
---> 34             return importlib.import_module('_amici')
     35     _amici = swig_import_helper()
     36     del swig_import_helper

/opt/conda/lib/python3.7/importlib/__init__.py in import_module(name, package)
    125                 break
    126             level += 1
--> 127     return _bootstrap._gcd_import(name[level:], package, level)
    128
    129

ModuleNotFoundError: No module named '_amici'
```

This shows kaggle does not have amici dependencies(SWIG, OPENBLAS etc) for direct installation of amici.

- Assuming version of packages could be an issue:

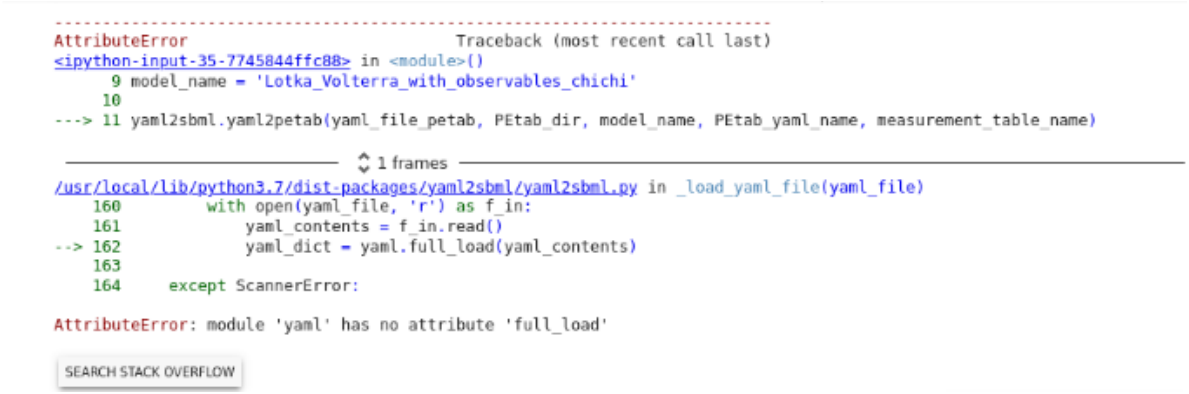
We updated pypesto, petab and amici but the error wasn't solved yet.
Package versions detail:
pypesto=0.0.13
petab=0.1.19

amici=0.11.17
yaml2sbml=5.19.0

Note: This implementation in colab could create a problem with the version of pandas that is being used. Colab needs pandas=1.1.0 while petab needs pandas=1.2.0, I switched to different versions of pandas when required.

- Assuming it was pyyaml version issue:

When installed pyyaml it was unable to generate problem.yml and getting full_load() attribute error (ref lotka volterra)



- Assuming it was collaborative error of PEtab and amici issue:

The reason for the error could be in reading in a PEtab problem from the yaml, that links the several PEtab-tsv tables+the SBML file. And since petab is dependent on amici, is it possible that either the amici package is not installed properly(I have not installed CBLAS) or any particular amici and PEtab package version were required. Checked with different combinations of amici and PEtab, but the error persisted.

- Assuming it was pypesto package version or its dependency issue:

On checking the git page, it did not show any pypesto dependencies like amici package and can be installed directly using **pip install pypesto** without considering the prerequisite of pypesto package(as there are none).

Checking Dependencies of pypesto and it were satisfied like below screenshot:

```
Requirement already satisfied: pypesto in /usr/local/lib/python3.7/dist-packages (0.2.6)
Requirement already satisfied: scipy>=1.5.2 in /usr/local/lib/python3.7/dist-packages (from pypesto) (1.7.0)
Requirement already satisfied: tqdm>=4.46.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (4.61.1)
Requirement already satisfied: cloudpickle>=1.5.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (1.6.0)
Requirement already satisfied: numpy>=1.19.1 in /usr/local/lib/python3.7/dist-packages (from pypesto) (1.19.5)
Requirement already satisfied: h5py>=3.1.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (3.1.0)
Requirement already satisfied: seaborn>=0.10.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (0.11.1)
Requirement already satisfied: pandas>=1.2.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (1.2.5)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.7/dist-packages (from pypesto) (3.4.2)
Requirement already satisfied: cached-property; python_version < "3.8" in /usr/local/lib/python3.7/dist-packages (from h5py>=3.1.0->pypesto) (1.5.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=1.2.0->pypesto) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=1.2.0->pypesto) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.3.0->pypesto) (7.1.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.3.0->pypesto) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.3.0->pypesto) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.3.0->pypesto) (0.10.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=1.2.0->pypesto) (1.15.0)
```

As all the pypesto dependencies are satisfied, we use SBML files with AmiciObjective() to perform the same task that we implemented above and encountered with the error.

```
model.requireSensitivitiesForAllParameters()

solver.setSensitivityMethod(amici.SensitivityMethod_forward)
solver.setSensitivityOrder(amici.SensitivityOrder_first)

objective = pypesto.AmiciObjective(model, solver, [edata], 1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-46-feadeb489239> in <module>()
      4 solver.setSensitivityOrder(amici.SensitivityOrder_first)
      5
----> 6 objective = pypesto.AmiciObjective(model, solver, [edata], 1)

 ---------------------------------------- 1 frames ----------------------------------------
/usr/local/lib/python3.7/dist-packages/pypesto/objective/amici_util.py in create_identity_parameter_mapping(amici_model, n_conditions
    103     x_ids = list(amici_model.getParameterIds())
    104     x_scales = list(amici_model.getParameterScale())
--> 105     parameter_mapping = ParameterMapping()
    106     for _ in range(n_conditions):
    107         condition_map_sim_var = {x_id: x_id for x_id in x_ids}

NameError: name 'ParameterMapping' is not defined
```

On running the objective declaration command, the received NameError, stating Parameter Mapping is not defined, was encountered.

Note: These trials to debug the error were performed on google colab notebook.

As the error count was undoubtedly increasing, a decision to carry forward further tasks on a local system was taken. There exists better tools that are specially designed for python language and create specific virtual environments for projects. PyCharm is one of the great tools that was encountered, where installing dependencies of packages is one of the easy tasks.

1.2 Installation and setup of PyCharm

PyCharm is an integrated development environment used in computer programming. This software supports easy installation of packages like amici due to its capacity to contain pre installed dependencies for various packages. One can install pycharm and create the project environment easily and start working on the project.
Click here to install pyCharm.

1.3 Errors in PyCharm and error handling

Errors are categorized in the following buckets:

1. Setting up the virtual env configuration

Check if you are accessing the packages and tools from the correct environment, to activate your virtual environment use this command : source ./bin/activate
This command will automatically turn on your virtual environment although you will see base activated as well along with the virtual environment.
(venv)(base)/home/documents/internship'21$

2. Installation and import packages

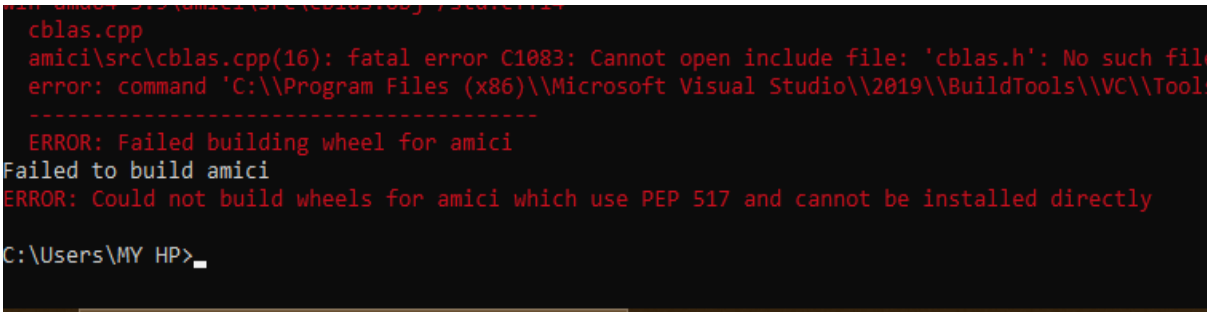
It depends in which environment do you choose to install your package into:
For frequently used packages like numpy and pandas, you can use sudo install numpy/pandas
For local use, specifically for a project, you may use pip install amici
For conda environment, you may install via conda install numpy(package-name)
You can set the environment you would want to work in while creating the project in PyCharm in the start itself.

3. Python version

There are packages that support a particular python version. For example, amici needs python>=3.7 as a dependency. You can set the python version you would want to use while creating the project in PyCharm in the start itself.

1.4 Example model in PyCharm and importing error(amici)

Implemented Boehm Model in PyCharm directly to have a proof check if pypesto is working fine and resulting in proper output. On loading up the yaml and measurement file in the created project, running the commands to create the PEtab file, amici error was encountered.



If we notice keenly, a red line can be observed with import amici.

To eliminate that, one may possibly use the following commands which potentially resolve the error.

```
sudo apt install libatlas-base-dev swig

# optionally for HDF5 support:
sudo apt install libhdf5-serial-dev
```

If your working environment is conda,

```
conda install -c conda-forge openblas
```

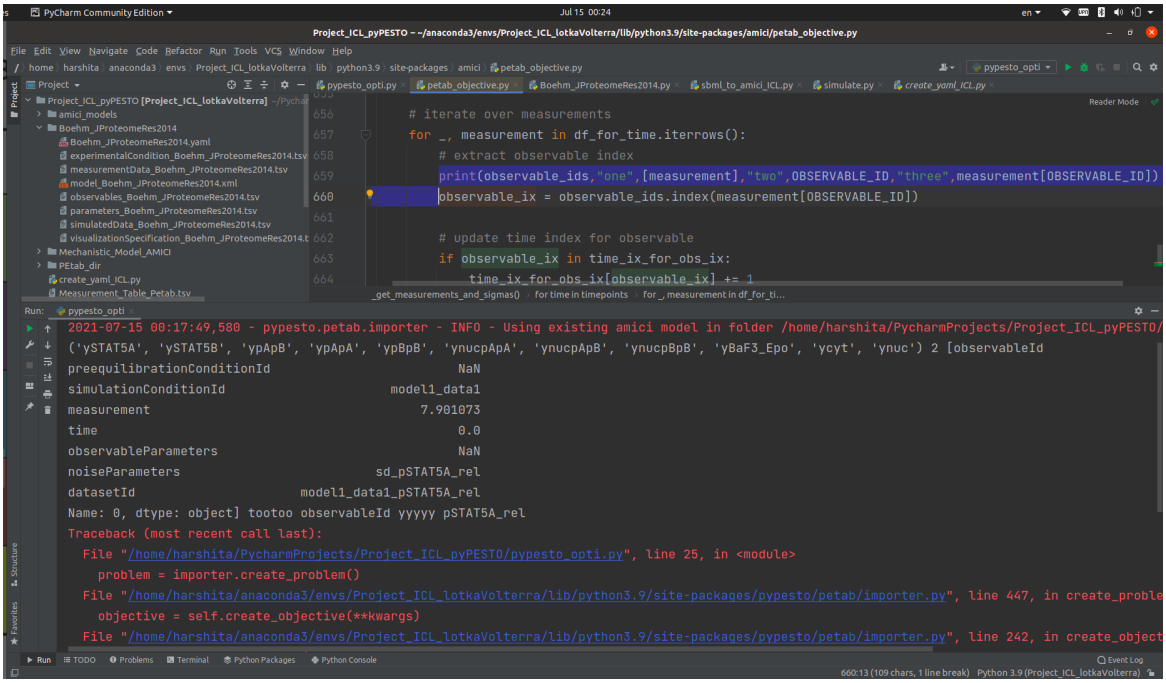
To install AMICI, now run:

```
pip install amici
```

One must uninstall amici first and try these commands.

1.5 Issues, errors and implementation of Lotka Volterra Model

PyCharm installation, setup, installation of necessary packages, and debugging package installation errors were all fixed. When running the extended yaml file with measurement file, encountered one more error while creating the objective function :



ValueError: tuple.index(x): x not in tuple while running problem = importer.create_problem()
https://github.com/ICB-DCM/pyPESTO/issues/701

This type of error generally occurs when the model is already defined(has petab file and amici models directory). It is suggested to delete them and run the model fresh again, which should solve the problem theoretically.

The error could not be debugged as a result of the discussion being done in this link. Took a break and decided to replicate the LV model in pycharm to clarify if the error is model based or embedded in PyCharm.

2. Lotka Voltera (LV)

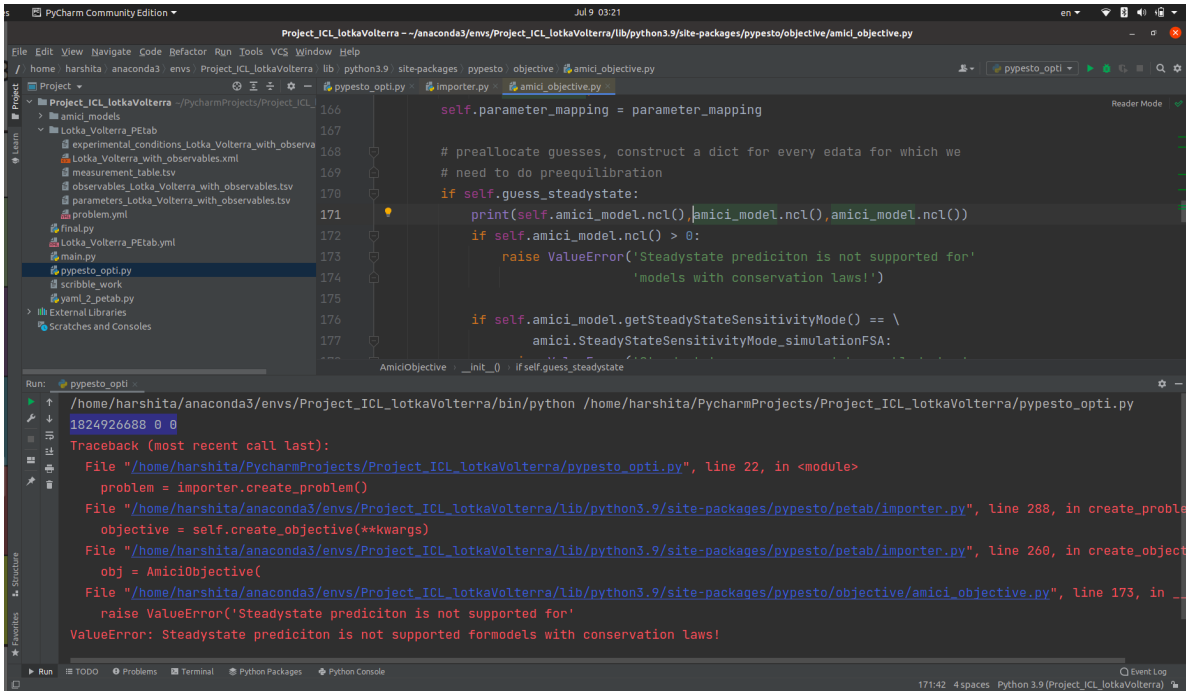
2.1 Workflow of implementation

- Firstly created a yaml file for model simulation in SBML that contains the **states, parameters** and as well as their initial values and dynamics.
- After this the yaml file was validated.
- Post validating the yaml file it was converted to SBML for model simulation, followed by SBML simulator to simulate the ODEs.
- After simulating the ODES it was used for conversion to PEtab for parameter fitting.
- For this purpose files like measurement file(.tsv) and petab files(.yml) were required fundamentally, that generated problem file(.yml), parameters, observables and conditionId files which were then used to implement the pypesto part.

2.2 Errors and debugging

While implementing and fitting these files pyPESTO got a tuple error, which wasted most of the time. Almost all the different methods were tried to remove the error (described in the previous section 1.1), but did not get debugged in colab. Finally moved to PyCharm, which seemed to have a lower error rate as compared to google colab or kaggle. Errors appeared in PyCharm could be handled and resolvable.

When creating the amici objective, following ValueError was encountered.
ValueError:Steady State prediction is not supported for models with conservation laws!



Updating the pypesto version, resolved it.
The command used to update pypesto: sudo apt-get update pypesto
Pypesto latest version used = 0.2.6
Raised github issues can be seen here.
Link: https://github.com/ICB-DCM/pyPESTO/issues/698

To visualize parameter fitting results by Lotka Volterra, when running this command, pypesto.visualize.parameters(result) Matplotlib.pyplot error had appeared:
“No output or plot for visualize.parameters(result)”.

To debug this, one can try installing matplotlib.pyplot and import it. Import matplotlib.pyplot as plt
Now run plt.show() after pypesto.visualize.parameters (result) command in the code. This shall solve the error.

To further explore the values of the optimized parameters, one can run this command:
optimized_parameters = result.optimize_result.get_for_key('x')

2.3 Output and results

The plot you get after running pypesto in the following manner:

Reference code that was compiled and run:

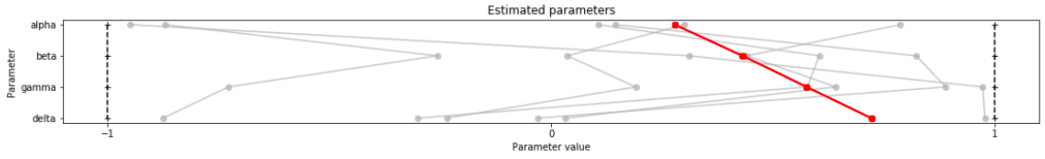
```
In [7]: %%capture
        import pypesto
        import pypesto.petab
        import pypesto.optimize as optimize
        import pypesto.visualize as visualize

        # import PEtab problem
        importer = pypesto.petab.PetabImporter.from_yaml(os.path.join(PEtab_dir, PEtab_yaml_name),
                                                         model_name=model_name)
        problem = importer.create_problem()
```

Perfomr the fitting:

```
In [8]: %%capture
        # perform optimization
        optimizer = optimize.ScipyOptimizer()
        result = optimize.minimize(problem,
                                   optimizer=optimizer,
                                   n_starts=10)
```

Next we want to visualize the results using parallel coordinate plots.

```
In [9]: visualize.parameters(result)
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff95ff93d30>
```



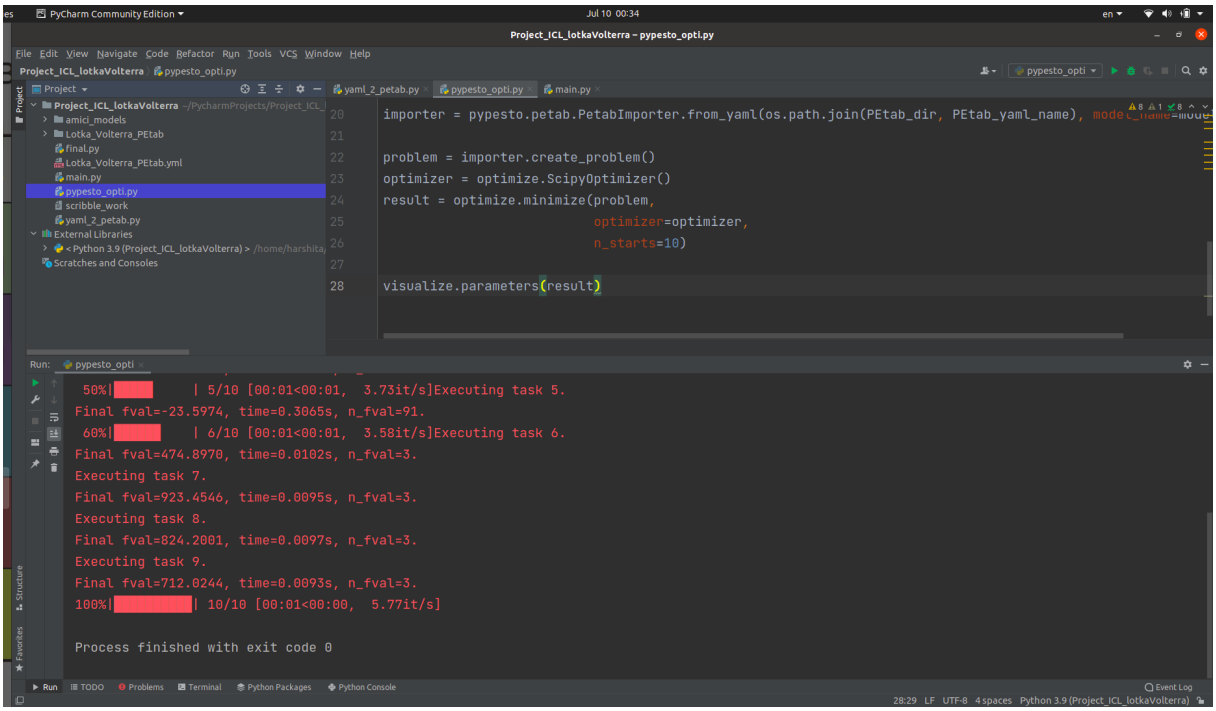


Figure 1: Successfully run code for 10 iterations to optimize parameters values

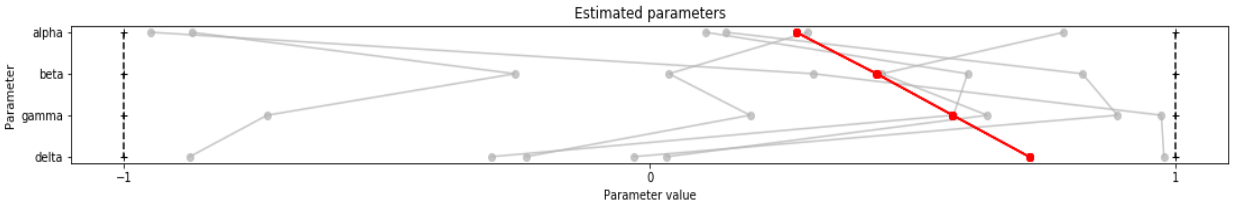


Figure 2 : Estimated value of the parameters by scipy optimizer(same as expected)

Now, that pypesto works completely fine with pycharm and can be used to run for the mechanistic model as well. We need to change the ODEs, parameters and data points in the measurement.tsv file.

3. Boehm Model

3.1 Why Boehm Model and workflow of implementation

Now the current need is to try different optimization methods like differential evolution to find the best fitted optimizer for a mechanistic model. To implement this, we tried replicating the Boehm model that resembles the mechanistic model closely(ODEs model type) and has been analysed for different optimizer methods.

For the implementation purpose files like measurement file(.tsv) and petab files(.yml) were required fundamentally that were used up from the Boehm_JProteomeRes2014 model folder directly.
In case, you want to generate these files from the basic yaml file, you may follow the approach listed below.

1. First a base folder in colab to store the files that were used while implementing the pypesto part.

```
!git clone --depth 1 https://github.com/Benchmarking-Initiative/Benchmark-Models-PEtab.git tmp/benchmark-models
|| (cd tmp/benchmark-models && git pull)

folder_base = "tmp/benchmark-models/Benchmark-Models/"
```

```
fatal: destination path 'tmp/benchmark-models' already exists and is not an empty directory.
Already up to date.
```

2. After creating the temp folder, import all the files of the Boehm_JProteomeRes2014 model into the same folder.

```
# a collection of models that can be simulated

#model_name = "Zheng_PNAS2012"
model_name = "Boehm_JProteomeRes2014"
#model_name = "Fujita_SciSignal2010"
#model_name = "Sneyd_PNAS2002"
#model_name = "Borghans_BiophysChem1997"
#model_name = "Elowitz_Nature2000"
#model_name = "Crauste_CellSystems2017"
#model_name = "Lucarelli_CellSystems2018"
#model_name = "Schwen_PONE2014"
#model_name = "Blasi_CellSystems2016"

# the yaml configuration file links to all needed files
yaml_config = os.path.join(folder_base, model_name, model_name + '.yaml')

# create a petab problem
petab_problem = petab.Problem.from_yaml(yaml_config)
```

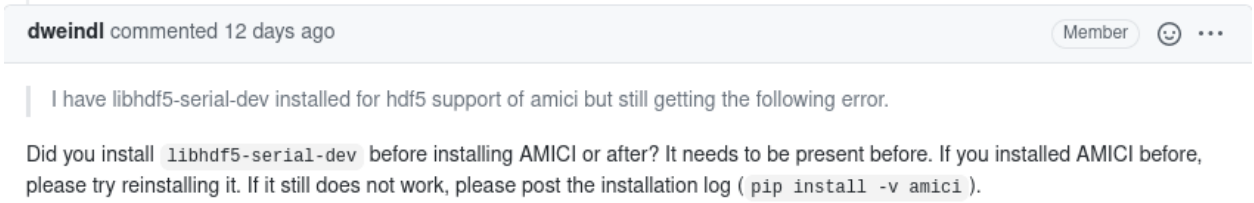
Now we can use petab_problem to implement pypesto for parameter optimization.

3.2 Errors, debugging and github raised issues

During the implementation part, when running pypesto, the error appeared at the following command:
problem = importer.create_problem()
The error encountered was,
Pickling the AmiciObjective requires an AMICI installation with HDF5 support

```
Traceback (most recent call last):
File "/home/harshita/PycharmProjects/Project_ICL_pyPESTO/pypesto_opti.py", line 55, in
result_powell = optimize.minimize(problem=problem,
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/pypesto/optimize/optimize.py", line 122,
in minimize
ret = engine.execute(tasks, progress_bar=progress_bar)
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/pypesto/engine/multi_process.py", line
58, in execute
pickled_tasks = [pickle.dumps(task) for task in tasks]
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/pypesto/engine/multi_process.py", line
58, in
pickled_tasks = [pickle.dumps(task) for task in tasks]
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/cloudpickle/cloudpickle_fast.py", line
73, in dumps
cp.dump(obj)
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/cloudpickle/cloudpickle_fast.py", line
563, in dump
return Pickler.dump(self, obj)
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/pypesto/objective/amici.py", line 234, in
getstate
amici.writeSolverSettingsToHDF5(
File "/home/harshita/anaconda3/envs/Project_ICL_lotkaVolterra/lib/python3.9/site-packages/amici/init.py", line 262, in
writeSolverSettingsToHDF5
amici.writeSolverSettingsToHDF5(_get_ptr(solver), file, location)
AttributeError: ("module 'amici.amici' has no attribute 'writeSolverSettingsToHDF5'", 'Pickling the AmiciObjective requires an AMICI
installation with HDF5 support.')
```

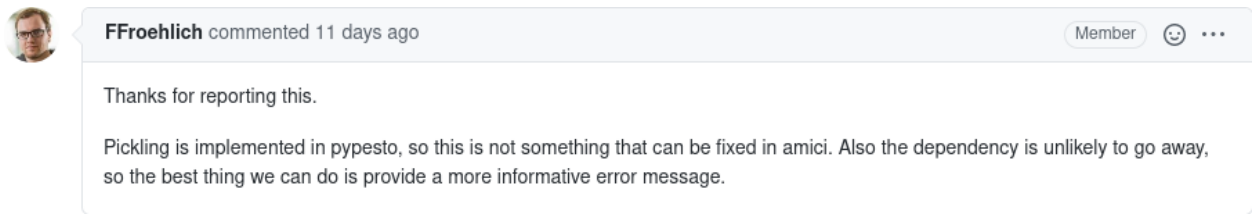
It was suggested to reinstall the amici package after installing the hdf5 package like below:



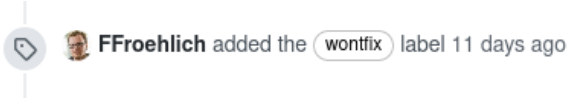
On uninstalling amici, checking the libhdf5-serial-dev package if it is installed, amici is installed again, but the error remains the same.

3.3 Inability to implement Boehm Model(dead end)

It was an amici problem and was suggested to raise this issue on AMICI-dev. The issue was raised on AMICI-dev, and the following response was recorded.



Finally it was assigned as a wontfix problem, and it can not be resolved.



Since the Boehm model cant be replicated now, to test the optimizers we extend the analysis done on Boehm to Lotka Volterra as a proof of concept. If that works we can directly use the mechanistic model ODEs, parameters and qualitative dataset generated via shared data spreadsheets.

To look at the detailed version of the error discussion, link to the raised github issues on pypesto and AMICI-dev is here.
PyPESTO: https://github.com/ICB-DCM/pyPESTO/issues/703
AMICI-dev: https://github.com/AMICI-dev/AMICI/issues/1533

4. Extended LV Model with Optimizers Analysis Done on Boehm Model

4.1 Workflow of implementation

1. Extended the Lotka Volterra after parameter optimization with more optimizers
2. Used L-BFGS-B and Powell gradient based optimization methods additionally to compare and find the best fitted optimizer

About L-BFGS-B and Powell optimization methods:

BFGS: It is a second-order optimization algorithm. It is a local search algorithm, intended for convex optimization problems with a single optima. L-BFGS-B - The method works by identifying fixed and free variables at every step (using a simple gradient method), and then using the L-BFGS method on the free variables only to get higher accuracy, and then repeating the process.
Powell's method: Strictly Powell's conjugate direction method, is an algorithm: for finding a local minimum of a function. The function need not be differentiable, and no derivatives are taken. The function must be a real-valued function of a fixed number of real-valued inputs.

PyPESTO provides a unified interface to a variety of optimizers of different types:
- All scipy optimizer (optimize.ScipyOptimizer(method=<method_name>))

[Implemented L-BFGS-B and Powell methods-gradient based optimizers]
  - function-value or least-squares-based optimizers
  - gradient or hessian-based optimizers

Code snippet to define different optimizers:

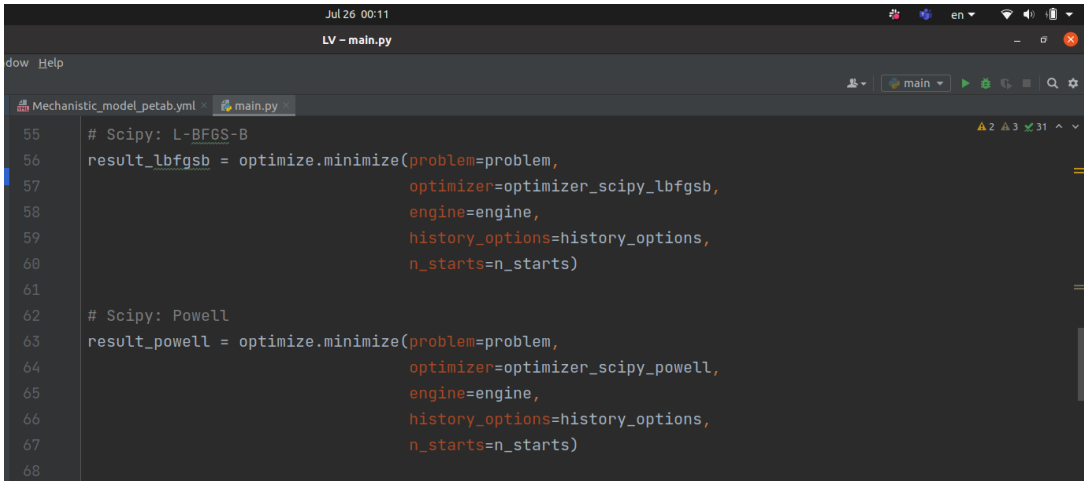
```
optimizer_scipy_lbfgsb = optimize.ScipyOptimizer(method='L-BFGS-B')
optimizer_scipy_powell = optimize.ScipyOptimizer(method='Powell')
```

The following performs 10 multi-start runs with different optimizers in order to compare their performance. The number of n_starts can be increased to 100 for more multi start runs. It is advisable only when having high RAM configuration else the functions below might take time in minutes to run.
Due to run time we use parallelization, and use the following functions.

```
engine = pypesto.engine.MultiProcessEngine()
history_options = pypesto.HistoryOptions(trace_record=True)
```

```
# Scipy: L-BFGS-B
result_lbfgsb = optimize.minimize(problem=problem,
                                  optimizer=optimizer_scipy_lbfgsb,
                                  engine=engine,
                                  history_options=history_options,
                                  n_starts=n_starts)

# Scipy: Powell
result_powell = optimize.minimize(problem=problem,
                                  optimizer=optimizer_scipy_powell,
                                  engine=engine,
                                  history_options=history_options,
                                  n_starts=n_starts)
```

Function to run optimizer that minimize the error:



4.2 Output and results

Figure of the optimized parameters result obtained with by default scipy optimizer

We can print the value of these parameters as well. The command to for this action is:
print(result.optimize_result.list[0])

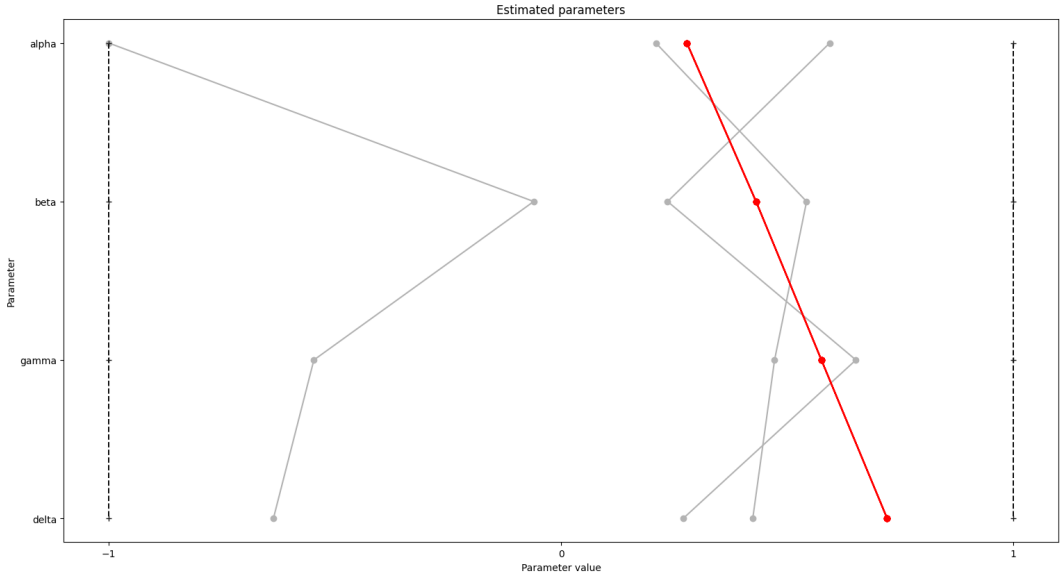


Figure 3 : Estimated value of the parameters by scipy optimizer(same as expected)

The waterfall plot obtained to compare the optimizers. The higher is the offsetted function value, the better is the optimizer. Methods like L-BFGS-B and Powell perform better due to being gradient based methods as compared to non-gradient based methods. In this case the L-BFGS-B method performs better than the Powell method as seen in the picture below.

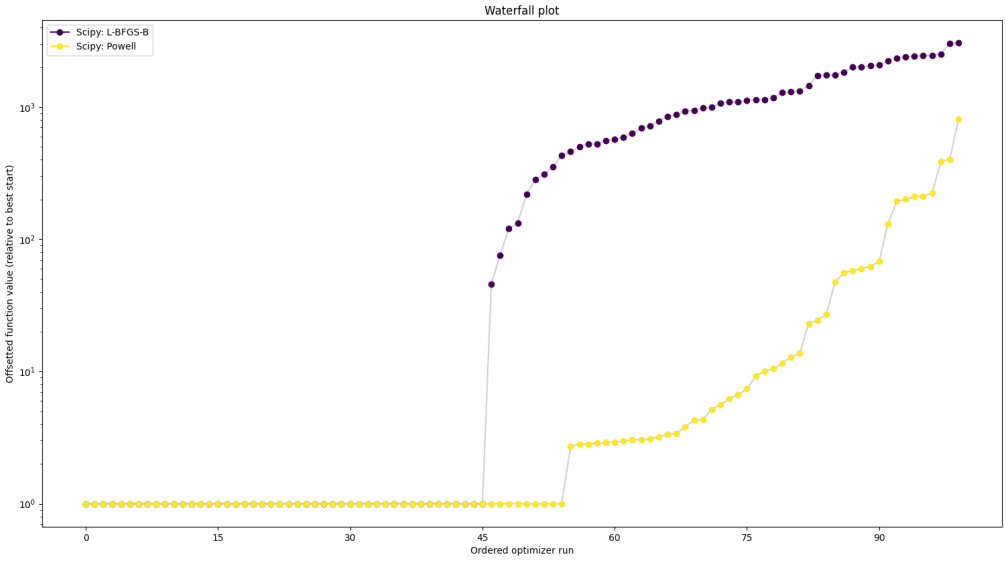


Figure 4 : Waterfall plot to compare two optimer methods wrt the number of iterations(how much better a model is wrt other optimizer methods)

PyPESTO focuses on two approaches to assess parameter uncertainties:

- Profile likelihoods
- Sampling

Profile likelihoods find confidence intervals via a test of likelihood ratio and on the parameter of interest it performs maximum-projection of the likelihood function. The likelihood ratio test then gives a cut off criterion via the chi-square distribution.
In pyPESTO, the maximum projection is solved as maximization problem and can be obtained via

```
%%time
%%capture

import pypesto.profile as profile

result = profile.parameter_profile(problem=problem,
                                   result=result,
                                   optimizer=optimizer_scipy_lbfgsb)
```

The maximum projections can now be inspected via:

```
# adapt x_labels..
x_labels = [f'Log10({name})' for name in problem.x_names]

visualize.profiles(result, x_labels = x_labels, show_bounds=True);
```

The plots obtained as a result of profile likelihood for Lotka Volterra:

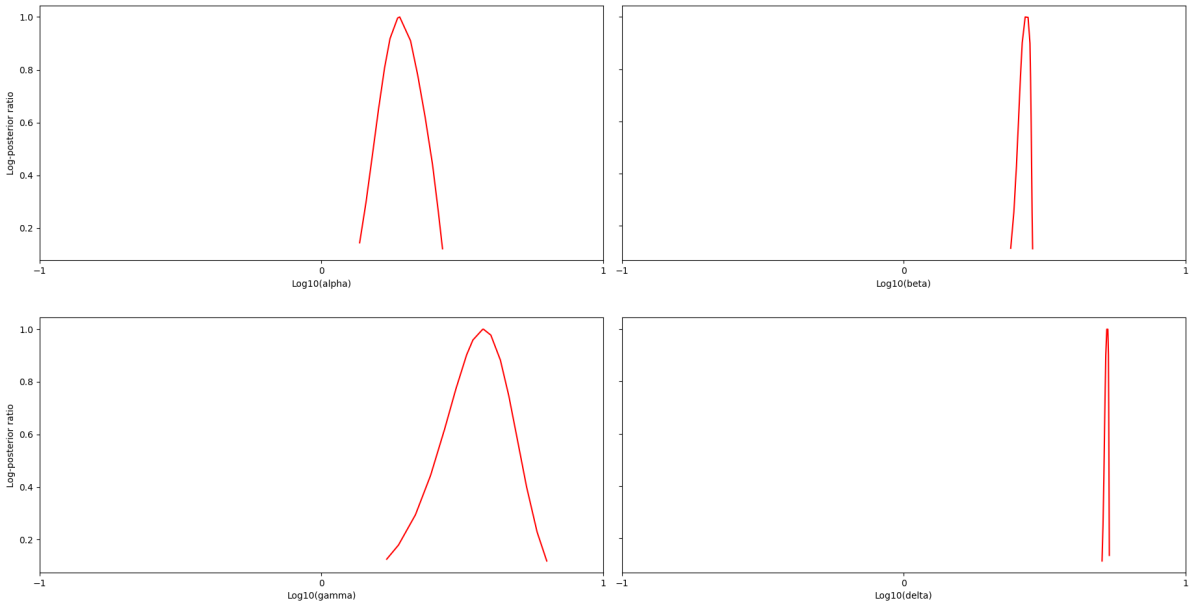


Figure 5 : Density vs log10(parameter) plot
(the peak density value is the most appropriate parameter value)

The plot shows that 4 parameters are identifiable, since the likelihood is tightly centered around the optimal parameter. One parameter (log10(gamma)) can be constrained by the data loosely.

Furthermore pyPESTO allows to visualize confidence intervals directly via

```
ax = pypesto.visualize.profile_cis(result, confidence_level=0.95, show_bounds=True)

ax.set_xlabel('Log10(Parameter value)');
```

The following plot was obtained showing the confidence interval of each parameter:

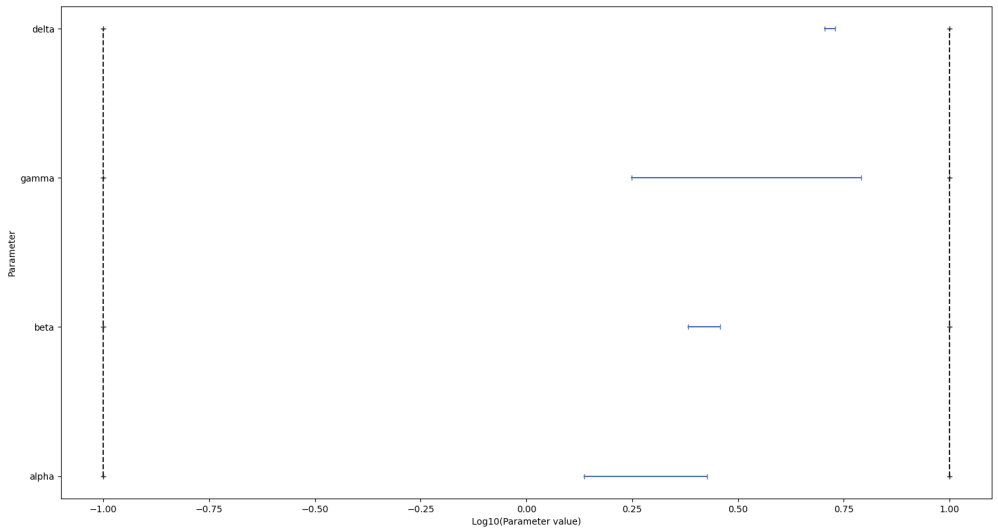


Figure 6 : Confidence intervals(range of values for parameters on a log10 scale)

Sampling

In pyPESTO, sampling from the posterior distribution can be performed as

```
import pypesto.sample as sample

n_samples = 10000

sampler = sample.AdaptiveMetropolisSampler()

result = sample.sample(problem,
                       n_samples=n_samples,
                       sampler=sampler,
                       result=result)
```

Gewekes test accesses convergence of a sampling run and computes the burn in of a sampling result. The effective sample size indicates the strenght the correlation between different samples.

sample.geweke_test(result=result)
result.sample_result['burn_in']

sample.effective_sample_size(result=result)
result.sample_result['effective_sample_size']

Visualization of Sampling Results

```
# scatter plots
ax = visualize.sampling_scatter(result)

# marginals
ax = visualize.sampling_1d_marginals(result)
```

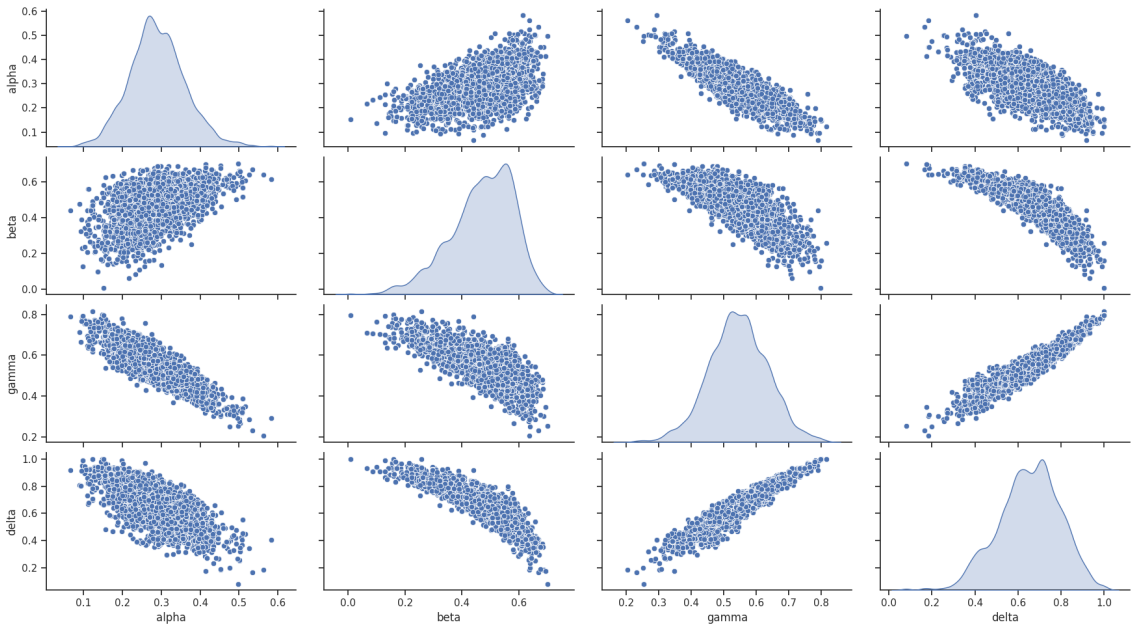


Figure 7 : Correlation plot of parameters

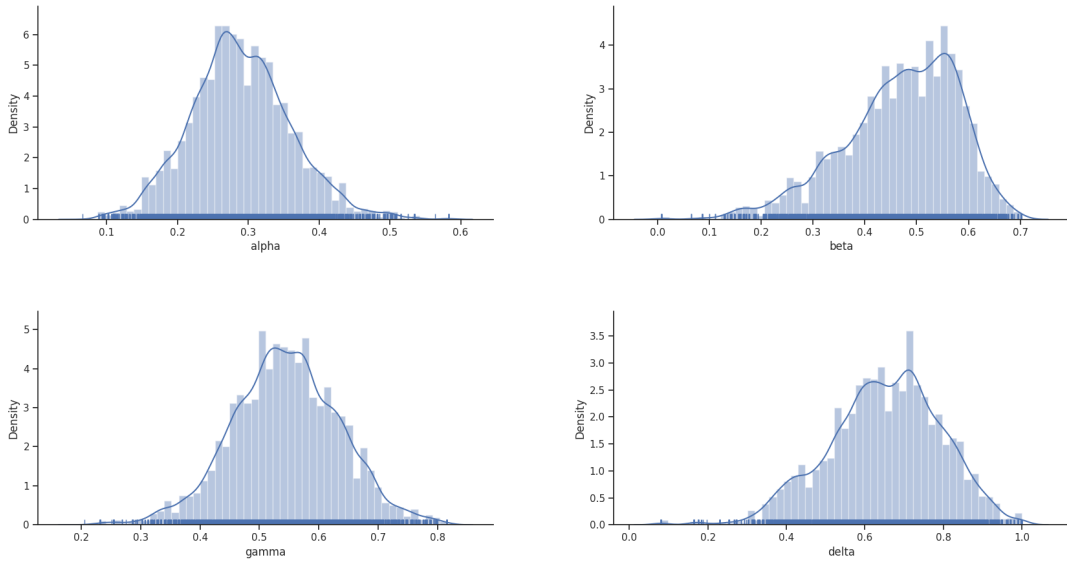


Figure 8 : Density vs Parameter value

The codes and the necessary files for the above plots can be found here

Sampler Choice:

Similarly to parameter optimization, pyPESTO provides a unified interface to several sampler/sampling toolboxes, as well as own implementations of sampler:

- Adaptive Metropolis: sample.AdaptiveMetropolisSampler()
- Adaptive parallel tempering: sample.ParallelTemperingSampler()
- Interface to pymc3 via sample.Pymc3Sampler()

Optimizer Choice, Optimizer Convergence, Uncertainty quantification (which were the uncommon parameter optimization analysis part between Lotka and Boehm) methods as an alternative to Boehm model. Because if this works, then we can change the Lotka files with mechanistic model files. We don't need the Boehm model now, and no need to worry about its errors.

5. Mechanistic model(pypesto)

5.1 Workflow of implementation

After getting with the optimizers used in the extended lotka part, we tried to use the same optimizers in the mechanistic model too. We already had all the PEtab files needed for pypesto implementation, the only thing we needed was the proper data file (measurement file).

5.2 Qualitative dataset preparation

To make the measurement.tsv file with the qualitative data we have, following assumptions were made and a small dataset was created.

a. Simplest model with only one condition ie condition1(can extend the dataset by providing different condition numbers for non-correlated conditions of experiment)
b. Can add more data points by adding multiple conditions ie condition1, condition2, condition3 etc
c. Did not distinguish between homogenate and balf as the data is limited (can be splitted if the data is more)
d. No initial fungal encounter consideration constraints (as the variables would change themselves accordingly) - can split them and check if any diff in value of parameters occurs though.
e. Considered only 5-8 weeks mice-say immunocompetent ones
f. Did not consider any vit D diet factor
g. t=0 to 48 hrs window (divided in the range of 0-2)
h. Log cfu/g for F variable, and logN for N
i. Intratracheal-respiratory toxicity
j. Cfu vs survival dataset used for F variable data points, TNF-alpha, IL-8, IL-12 were used to create data points for cytokines C, and Neutrophils for N
k. Choose the section that has highest number of datapoints in the selected area (constraints like age, mode of getting infection etc) for condition1

	K	L	M	N	O	P	Q	R
8	ATCC 13073	ATCC 13073	Af293	CEA10	from IA patient	from IA patient	ATCC 13073	ATCC 13073
9	Murine	Murine	Murine	Murine	Murine	Murine	Murine	Murine
10	–	BALB/c	C57BL/6	C5BL/6J	C57BL/6J	C57BL/6J	BALB/c	BALB/c
11	–	8-13 weeks	age-matched	8-10 weeks	12 weeks	12 weeks	6-8 weeks	5-8 weeks
12	RB6-8C5 mAb +doxy	–	–	–	Vit D in diet	No vit D in diet	–	Sepsis
13	intratracheal	–	–	–	–	–	–	–
14	BALF	intratracheal	i.t.	i.t.	intratracheal	intratracheal	intra-tracheal	intratracheal
15	–	BALF	BALF	BALF	BALF	BALF	BALF	BALF
16	–	100 ul	–	2 ml	3.2 ml	3.2 ml		–
17	4-7 mice	5 mice	11-14 mice	3-5 mice	5 mice	5 mice	3-4 mice	4-8 mice
18	Flow cytometry	Cytospin	Flow cytometry	Cytospin	Cytospin	Cytospin	Flow cytometry	–
19	Neutrophil numbers in times after DOX	Neutrophils (x10^6)	Neutrophil counts (x10^5)	Neutrophil counts (x10^6)	Neutrophil (%)	Neutrophil (%)	Neutrophil counts (x10^4)	Neutrophil counts (x10^6)
20	125	0.003375	–	–	0.25	0.25	–	–
21	–	–	–	–	–	–	–	–
22	–	–	–	–	–	–	–	–
23	–	0.08	–	–	–	–	–	–
24	–	–	–	–	–	–	–	0
25	–	0.75	–	–	50	58	–	–
26	–	–	–	–	–	–	–	–
27	–	0.7	–	–	–	–	–	–
28	–	–	0.5	–	–	–	–	–
29	–	1	–	0.8	85	85	10	–
30	–	–		–	–	–	–	–
31	–	–	–	–	–	–	–	–
32	1125	1	–	1.25	83	86	25	0.5

Neutrophiles Paper screening

	A	B	C	D	E	F	G	H	I
1	**A. fumigatus strain**	ATCC 201795	Af293-GFP	Clinical isolate	ATCC 13073	ATCC 13073	Clinical isolate	ATCC 201795	Clinical isolate
2	**Infection model**	Murine	Murine	Murine	Murine	Murine	Murine	Murine	Murine
3	**Strain**	Crl:CF1	c57BL/6	CD1	C57BL/6	–	CD1	Crl:CF1	C57BL/6
4	**Age**	11 - 13g	8-11 weeks	6-8 weeks	6-8 weeks	–	6-8 weeks	11 - 13g	7 weeks
5	**Treatment**	–	–	–	100 ug RB6-8C5	RB6-8C5	Cortisone acetate	–	Neutropenia by vinbla (5mg/kg)
6	**Cell concentration**	–	–	–	–	–	–	–	–
7	**Route of infection**	inhalation chamber	Inhalation chamber (90 s)	Intranasal	Intratracheal	Intratracheal	Intranasal	inhalation chamber	Intratracheal
8	**Sample**	Homogenate	Homogenate	Homogenate	Homogentate	Homogenate	Homogenate	BALF	BALF
9	**Sample volume**	10ml	2 ml	–	50ul	–	–	1.6 ml	4 ml
10	**Sample size**	5 mice	8-12 mice	5 mice	5 mice	5 mice	5 mice	5 mice	3 mice
11	**Analysis**	ELISA	ELISA	PCR	ELISA	ELISA	PCR	ELISA	ELISA
12	**Time**	pg/ml	pg/ml	IL-12p40 Arbitrary units	pg/ml	pg/ml	IL-12p40Arbitrary units	pg/ml	pg/ml
13	**0**	5	450	0	870		0.3	5	–
14	**1**		500	–		–	–	–	–
15	**2**	3	–	–	–	–	–	0	–
16	**3**		–	–	–	–	–	–	–
17	**4**	2	–	–	–	–	–	0	–
18	**5**		–	–			–	–	–
19	**6**		600	–	–	–	–	–	–
20	**8**		–	–	–	–	–	–	–
21	**9**		–	–	–	–	–	–	–
22	**12**	2	–	–	–	–	–	10	–
23	**18**		–	–	–	–	–	–	–
24	**24**	26	750	0.6	–	–	0.2	95	120
25	**30**		–	–			–	–	–

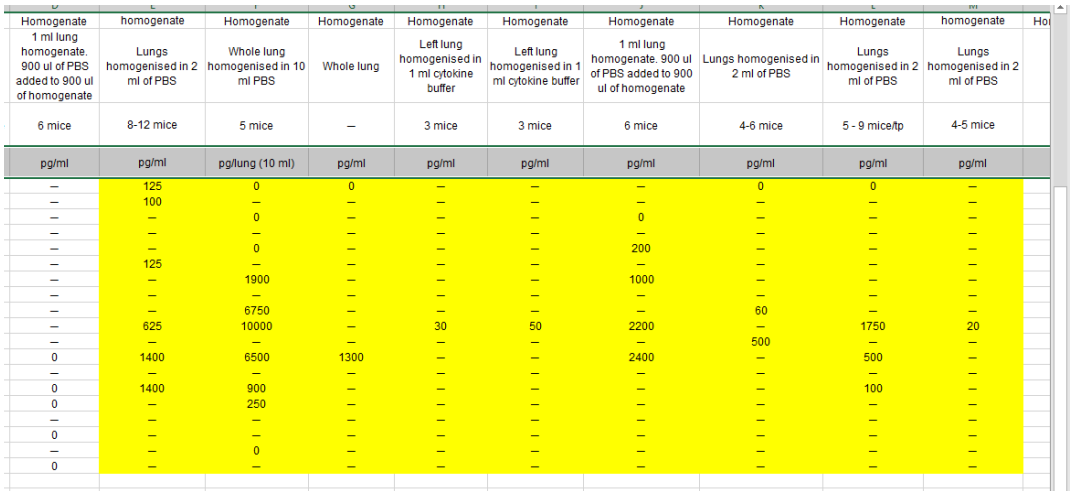
IL-12

11	**Strain**	C57BL/6	C57BL/6J	BALB/cf C57BL/6	C57BL/6	C57BL/6	C57BL/6	BALB/c	C57BL/6	129/SvEv
12	**Age**	6-8 weeks	8-11 weeks	18-22 g	Age-matched	age-matched	8-10 weeks	6-8 weeks	8 weeks (20g)	6-8 weeks
13	**Treatment**	–	–	–	–	–	ctrl Ab	–	–	–
14	**Route of infection**	intra-tracheal	inhalation (chamber)	intranasal	intra-tracheal	intratracheal	intra-tracheal	intra-tracheal	intra-tracheal	intratracheal
15	**Sample**	homogenate	homogenate	Homogenates	Homogenate	Homogenate	homogenate	homogenate	Homogenate	Homogenate
16	**Sample volume**	1 ml lung homogenate. 900 ul of PBS added to 900 ul of homogenate	Lungs homogenised in 2ml PBS	Homogenised in 1ml PBS/100mg tissue	Lungs homogenised in 2 ml of PBS	Lungs homogenised in 2 ml of PBS	Lungs homogenised in 2 ml of PBS	Right lung homogenised in PBS	Left lung homogenised in PBS + protease inhibitor	Right lung homogenised in PBS
17	**Sample**	6 mice	8-15 mice	3 mice	5-7 mice	5 - 9 mice	4-5 mice	4-6 mice	5 mice	5 mice
18	**Time**	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml
19	**0**	0	0	–	100	250	–	–	–	–
20	**1**	–	100	–	–	–	–	–	–	–
21	**2**	–	–	–	–	–	–	–	–	–
22	**3**	–	–	–	–	–	–	–	–	–
23	**4**	–	–	–	–	–	–	–	–	–
24	**6**	–	1100	–	–	–	–	–	–	–
25	**8**	–	–	–	–	–	–	–	–	–
26	**9**	–	–	–	–	–	–	–	–	–
27	**10**	–	–	–	450	–	–	–	–	–
28	**12**	–	–	–	–	–	–	–	–	–
29	**24**	2500	2000	750	–	750	590	2900	–	2200
30	**36**	–	–	–	450	–	–	–	–	–
31	**48**	1000	1600	95	–	500	–	6000	–	–
32	**60**	–	–	–	–	–	–	–	–	–
33	**72**	–	1250	–	–	250	–	–	300	–
34	**96**	0	–	–	–	–	–	–	–	–

IL-8

Treatment	–	–		Treatment	–	Vit D in diet		Treatment	No vit D in di
Route of infection	inhalation (chamber)	Intra-tracheal		Route of infection	Intra-tracheal	Intra-tracheal		Route of infection	Intra-trache
Time (hours)	CFUs (x10^6)	CFUs (x10^6)		Time (hours)	log CFU/g	log10CFU/g		Time (hours)	log10CFU/
0	–	–		0	–	–		0	–
1	–	–		1	–	–		1	–
2	–	–		2	–	–		2	–
3	–	–		3	–	–		3	–
4	–	–		4	7.2	–		4	–
6	–	–		6	–	–		6	–
8	–	–		8	–	–		8	–
9	–	–		9	–	–		9	–
10	–	–		10	–	–		10	–
12	–	–		12	6.8	–		12	–
16	–	–		16	–	–		16	–
18	–	–		18	–	–		18	–
24	2.1	–		24	6.4	5.6		24	9
36	–	–		36	–	–		36	–
40	–	–		40	–	–		40	–
48	–	–		48	4.3	4.5		48	7
60	–	–		60	–	–		60	–
72	1.5	–		72	3.8	3.5		72	5
96	–	0.2		96	2.5	–		96	–
120	–	–		120	–	–		120	–
144	–	–		144	–	–		144	–
168	–	–		168	–	0		168	0

Survival vs CFU(For Fungal concentration F)



Homogenate	homogenate	Homogenate	Homogenate	Homogenate	Homogenate	Homogenate	Homogenate	Homogenate	homogenate	Ho
1 ml lung homogenate. 900 ul of PBS added to 900 ul of homogenate	Lungs homogenised in 2 ml of PBS	Whole lung homogenised in 10 ml PBS	Whole lung	Left lung homogenised in 1 ml cytokine buffer	Left lung homogenised in 1 ml cytokine buffer	1 ml lung homogenate. 900 ul of PBS added to 900 ul of homogenate	Lungs homogenised in 2 ml of PBS	Lungs homogenised in 2 ml of PBS	Lungs homogenised in 2 ml of PBS	
6 mice	8-12 mice	5 mice	–	3 mice	3 mice	6 mice	4-6 mice	5 - 9 mice/tp	4-5 mice	
pg/ml	pg/ml	pg/lung (10 ml)	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	pg/ml	
–	125	0	0	–	–	–	0	0	–	
–	100	–	–	–	–	–	–	–	–	
–	–	0	–	–	–	0	–	–	–	
–	–	–	–	–	–	–	–	–	–	
–	–	0	–	–	–	200	–	–	–	
–	125	–	–	–	–	–	–	–	–	
–	–	1900	–	–	–	1000	–	–	–	
–	–	–	–	–	–	–	–	–	–	
–	–	6750	–	–	–	–	60	–	–	
–	625	10000	–	30	50	2200	–	1750	20	
–	–	–	–	–	–	–	500	–	–	
0	1400	6500	1300	–	–	2400	–	500	–	
–	–	–	–	–	–	–	–	–	–	
0	1400	900	–	–	–	–	–	100	–	
0	–	250	–	–	–	–	–	–	–	
–	–	–	–	–	–	–	–	–	–	
0	–	–	–	–	–	–	–	–	–	
–	–	0	–	–	–	–	–	–	–	
0	–	–	–	–	–	–	–	–	–	

TNF alpha in vivo data

Figure 9 : Different Data Files

File Edit Format View Help

observableId	simulationConditionId	measurement	time	noiseParameters
fungal_conc	condition1	7.2	0.166666666666667	0.1
fungal_conc	condition1	6.8	0.5	0.1
fungal_conc	condition1	6.4	1	0.1
fungal_conc	condition1	4.3	2	0.1
cytokines_tnf	condition1	0	0.083333333333333	0.1
cytokines_tnf	condition1	200	0.166666666666667	0.1
cytokines_tnf	condition1	1000	0.333333333333333	0.1
cytokines_tnf	condition1	2200	1	0.1
cytokines_tnf	condition1	2400	2	0.1
neutrophils_conc	condition1	5	0.5	0.1
neutrophils_conc	condition1	5.39794	1	0.1
neutrophils_conc	condition1	5.47712	2	0.1

Figure 10 : Formatted dataset

Now we are all set to run pypesto with optimizers and test the optimized parametric values.

5.3 Errors and debugging

While running pypesto, at importer.create_problem() step AttributeError was encountered.

free_syms = sorted(sp.sympify(formula).free_symbols,
AttributeError: 'function' object has no attribute 'free_symbols'

Github link for the raised issue: https://github.com/ICB-DCM/pyPESTO/issues/703

To debug this error, change N to NN: there is a sympy function sympy.core.evalf.N interfering with that. As a workaround you can just go for 'NN' then. Needs to be fixed in PEtab.

5.4 Gradient based optimization methods using PyPESTO

Now running pypesto on petab file and measurement.tsv, following results were obtained.

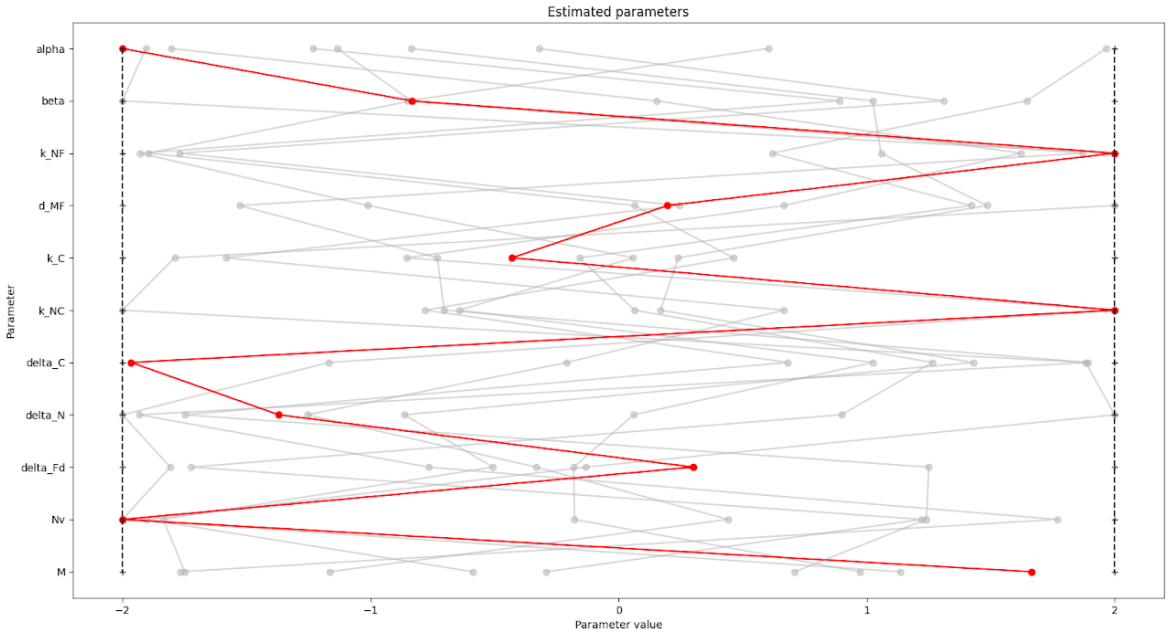


Figure 11 : Estimated parameter values for Scipy default optimizer

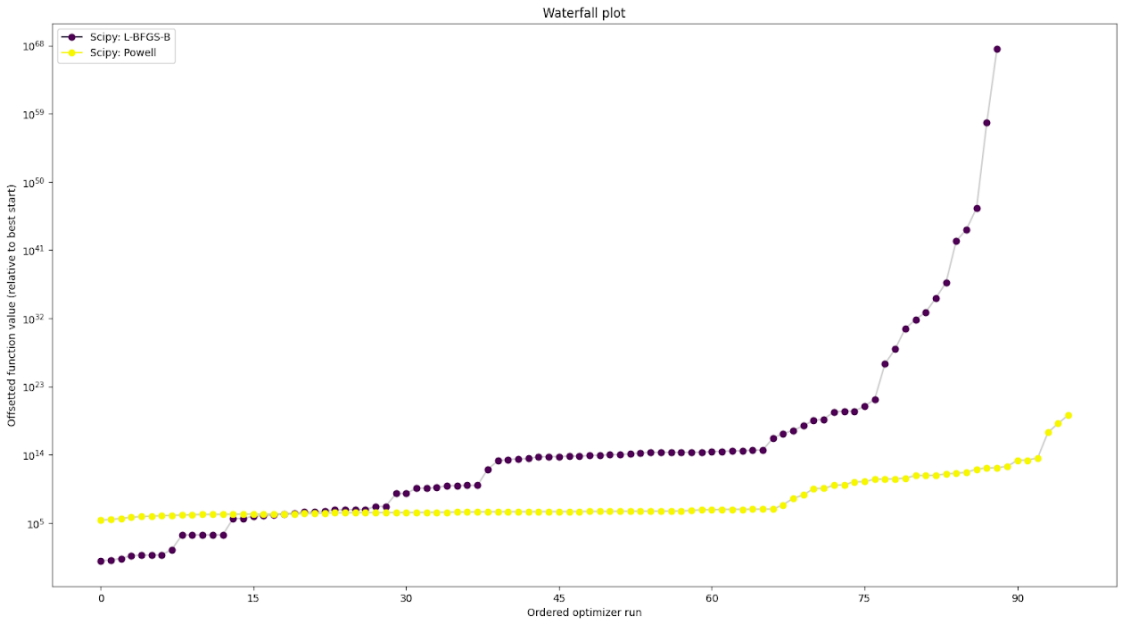


Figure 12 : Waterfall plot for L-BFGS-B and Powell

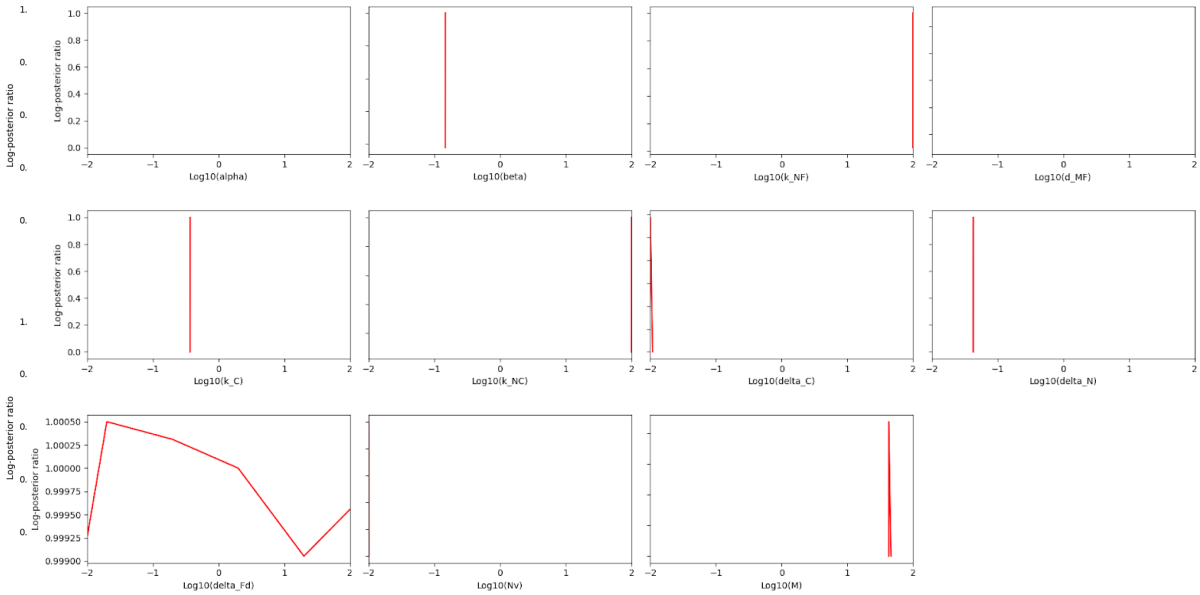


Figure 13 : Density vs log10(parameter) plot

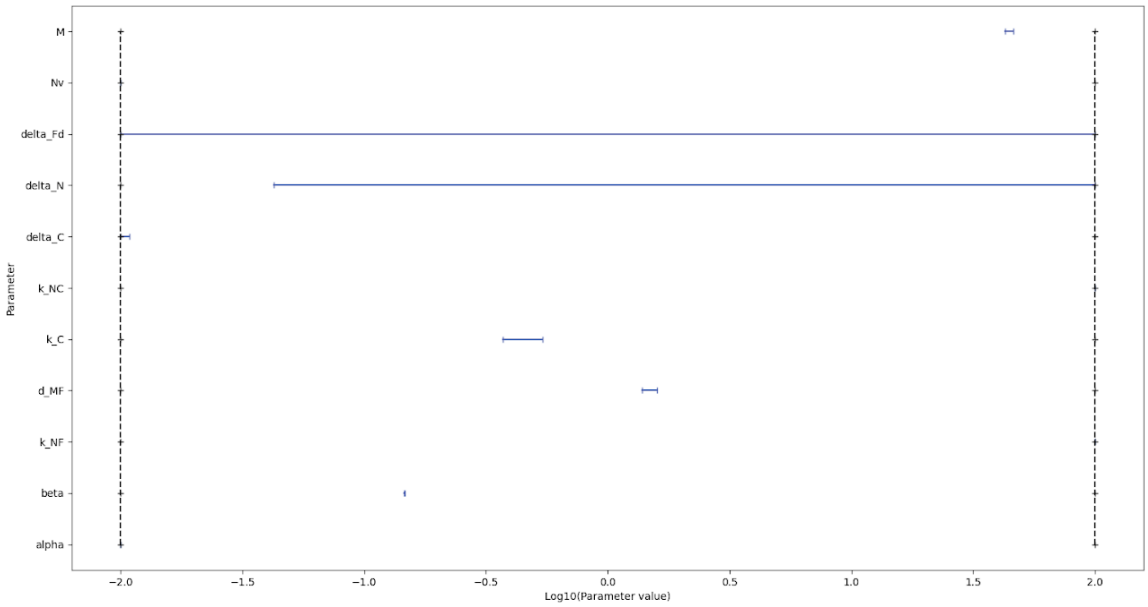


Figure 14 : Confidence intervals(range of values for parameters on a log10 scale)

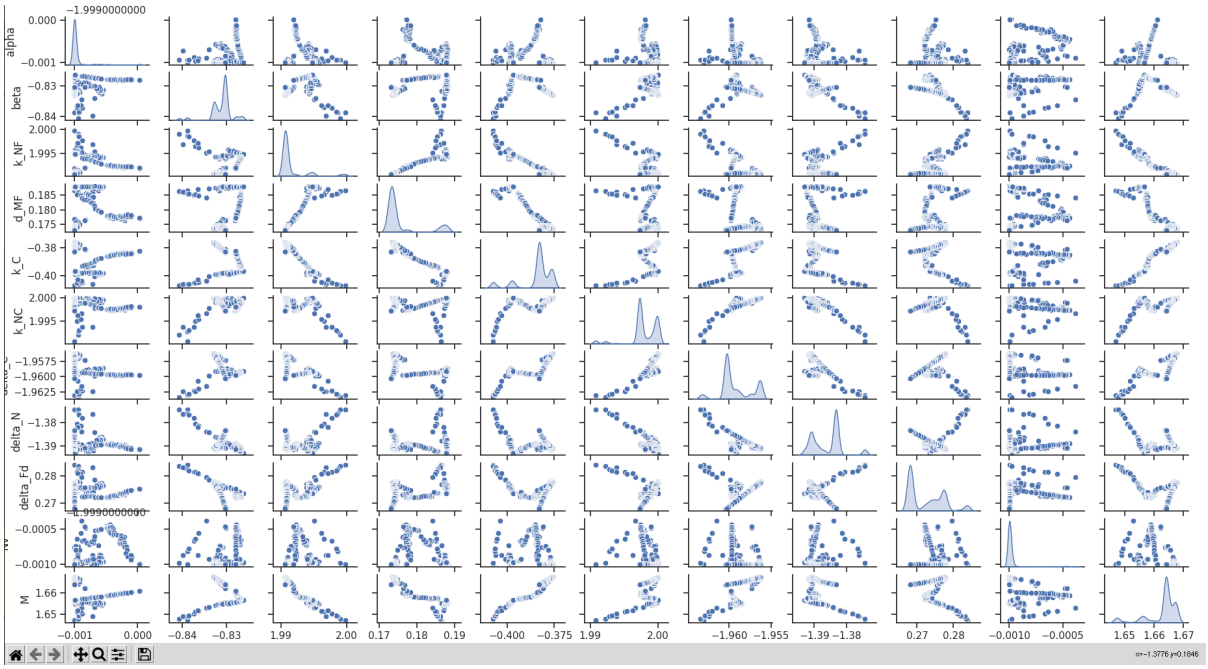


Figure 15 : Correlation plot of parameters

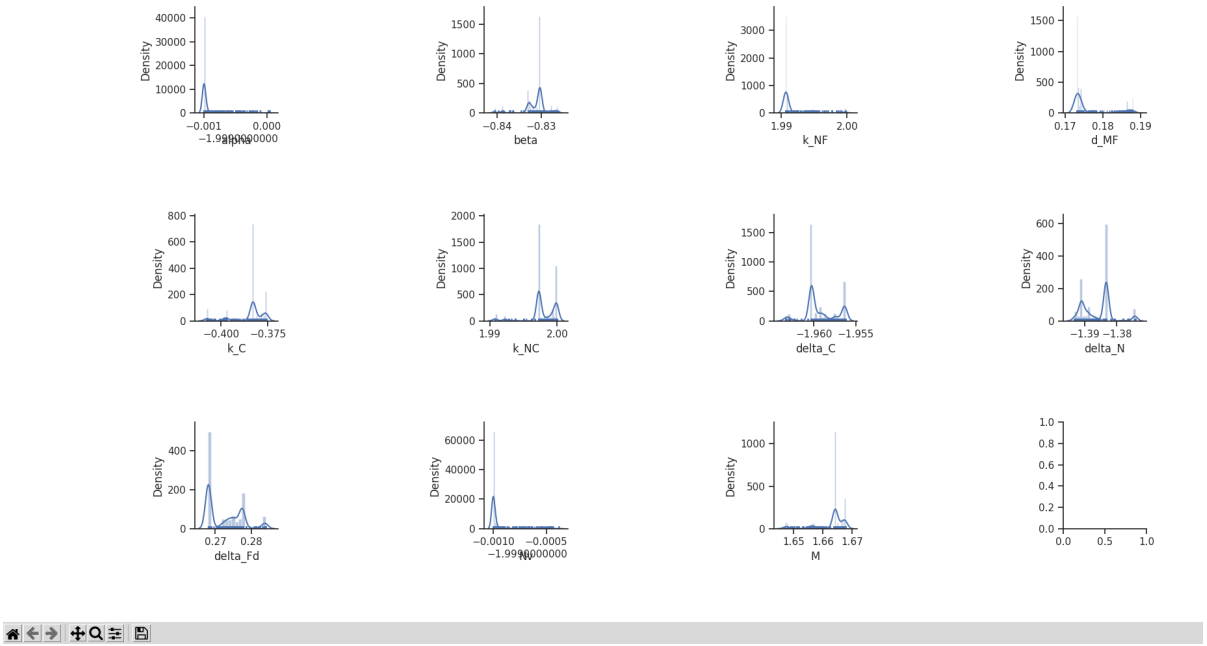


Figure 16 : Density vs Parameter value

5.5 Non-gradient(gradient free) based optimization methods using PyPESTO

Now we have more optimization methods. We have analysed gradient based methods above. Here we implement some non gradient based methods.

- **Differential evolution (DE)** is a population-based metaheuristic search algorithm that optimizes a problem by iteratively improving a candidate solution based on an evolutionary process. Such algorithms make few or no assumptions about the underlying optimization problem and can quickly explore very large design spaces.

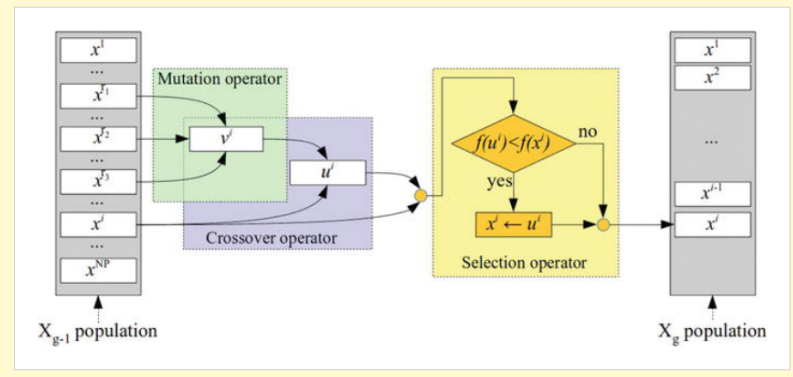


Figure 17 : Visual interpretation of Differential Evolution algorithm

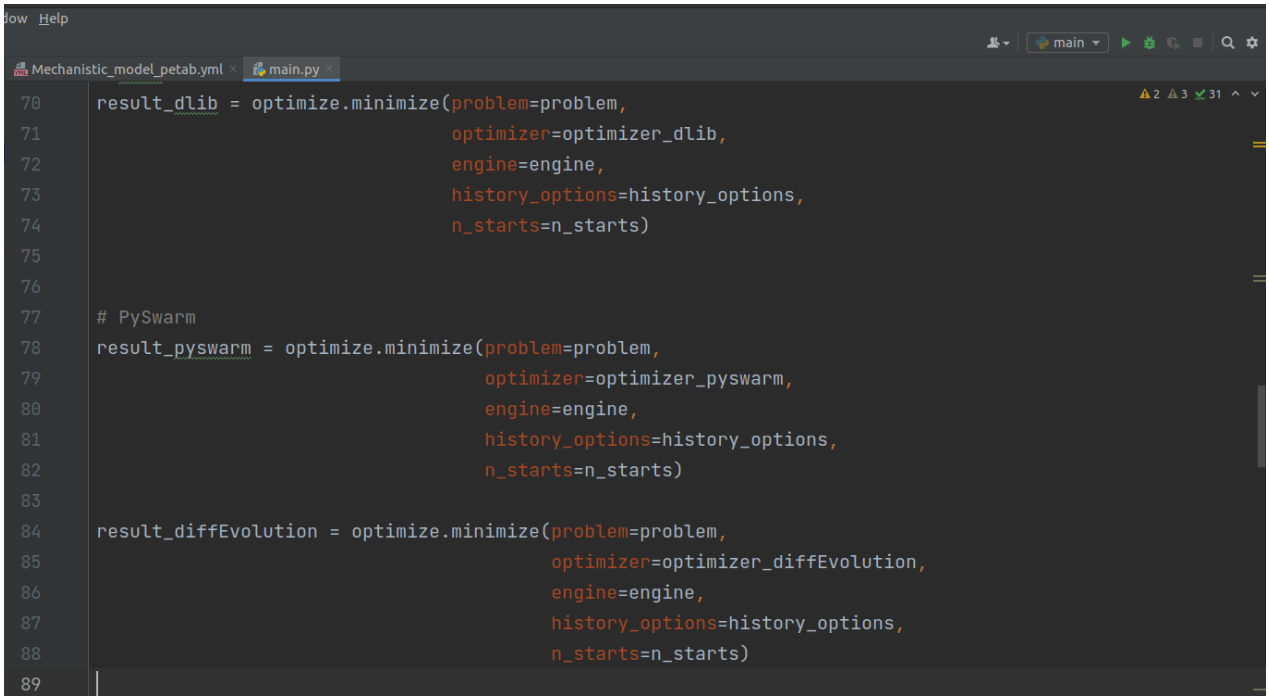
Ref: Rafael Rivera-Lopez and Juana Canul-Reich, “Differential Evolution Algorithm in the Construction of Interpretable Classification Models”

- **Dlib:** DLib implements a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction. It can be used in the python program using pip install dlib.
- **Pyswarm**: PySwarms is an extensible research toolkit for particle swarm optimization (PSO) in Python. The basic PSO is influenced by a number of control parameters, namely the dimension of the problem, number of particles, acceleration coefficients, inertia weight, neighbor- hood size, number of iterations, and the random values that scale the contribution of the cognitive and social components. It is intended for a high-level declarative interface for implementing PSO in the problems. It can be installed in the code by using pip install pyswarm.

- Differential Evolution(optimize.ScipyDifferentialEvolutionOptimizer())
- Dlib (optimize.DlibOptimizer(options={'maxiter': <max. number of iterations>}))
    - Global optimizer
    - Gradient free
- Particle Swarm (optimize.PyswarmOptimizer())
    - Particle swarm algorithm
    - Gradient free

To define and run these optimizers in pypesto, the command used was:

- optimizer_dlib = optimize.DlibOptimizer()
- optimizer_pyswarm = optimize.PyswarmOptimizer()
- optimizer_diffEvolution = optimize.ScipyDifferentialEvolutionOptimizer()



Now running this, gives the following warning and takes a huge time to run. The mentioned warning keeps coming and the program doesnt terminate.
[Warning] AMICI:CVODES:CVode:TOO_MUCH_WORK: AMICI ERROR: in module CVODES in function CVode :
At t = 0.876855, maximum steps taken before reaching out.

[Warning] AMICI:simulation: AMICI forward simulation failed at t = 0.876855:
AMICI failed to integrate the forward problem.

This warning shows up when readSimulationState tries to set an empty state. It shouldn't be an issue as we have discrete data availability.

The time taken by the code can be fixed by changing the n_starts variable value from 100 to 10 or a lower number(not suggested lower than 10) and max_iters = 100(10 is too less and 1000 would be suggested if the system has GPU support) parameter in the optimizer declaration.
Hence the declaration of optimizers become :

```
optimizer_scipy_lbfgsb = optimize.ScipyOptimizer(method='L-BFGS-B',options={'maxiter': 100})
optimizer_scipy_powell = optimize.ScipyOptimizer(method='Powell',options={'maxiter': 100})
optimizer_dlib = optimize.DlibOptimizer(options={'maxiter': 100})
optimizer_pyswarm = optimize.PyswarmOptimizer(options={'maxiter': 100})
optimizer_diffEvolution = optimize.ScipyDifferentialEvolutionOptimizer(options={'maxiter': 100})
```

This takes finite time to optimize parameters in different optimizers supported by pypesto.
This error can also be fixed by setting the **setMaxSteps variable**. This information can be found in detail at the link below.
https://github.com/AMICI-dev/AMICI/issues/1535

Plots:
For n_starts = 10

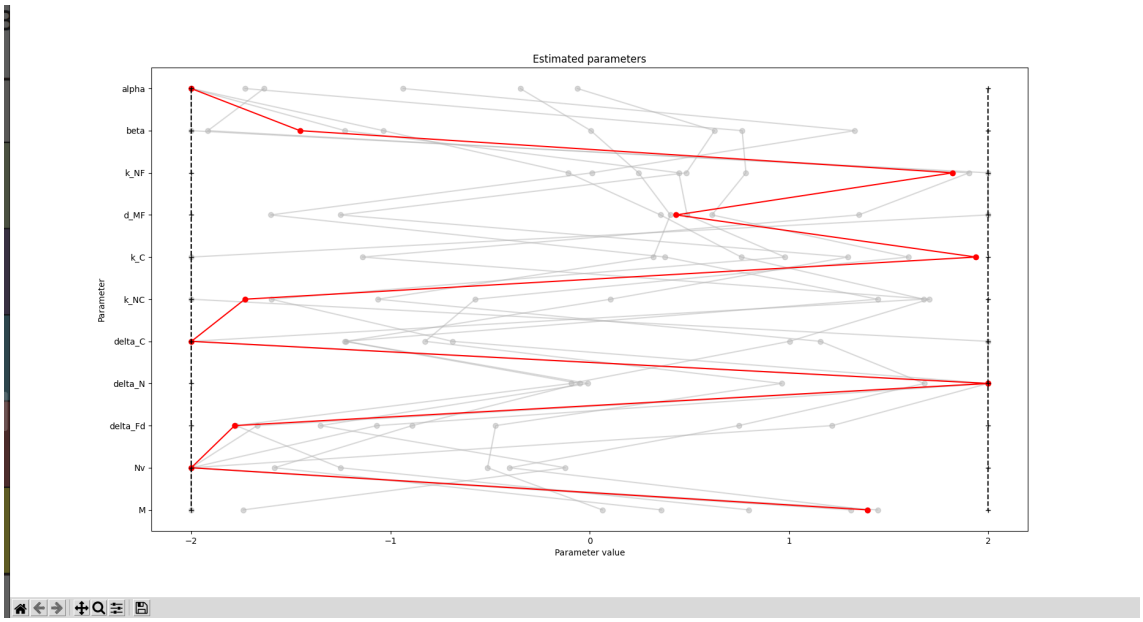


Figure 18 : Scipy default optimizer plot for parameters value: (max_iter: default)

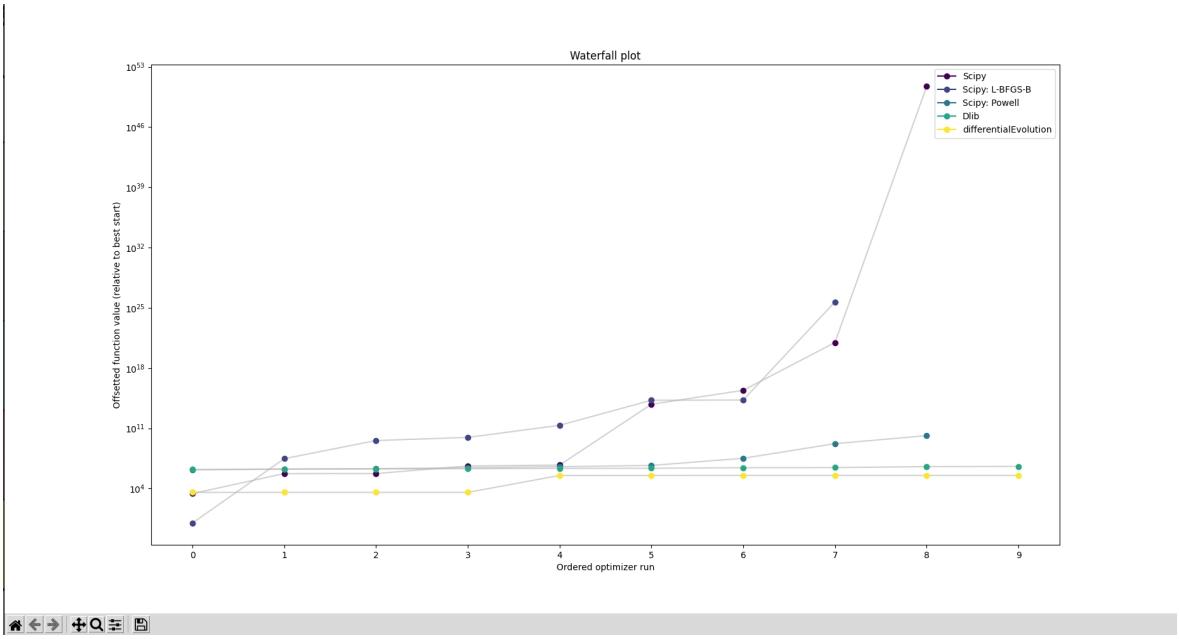


Figure 19 : Offsetted value of different optimizers

For max_iter=100

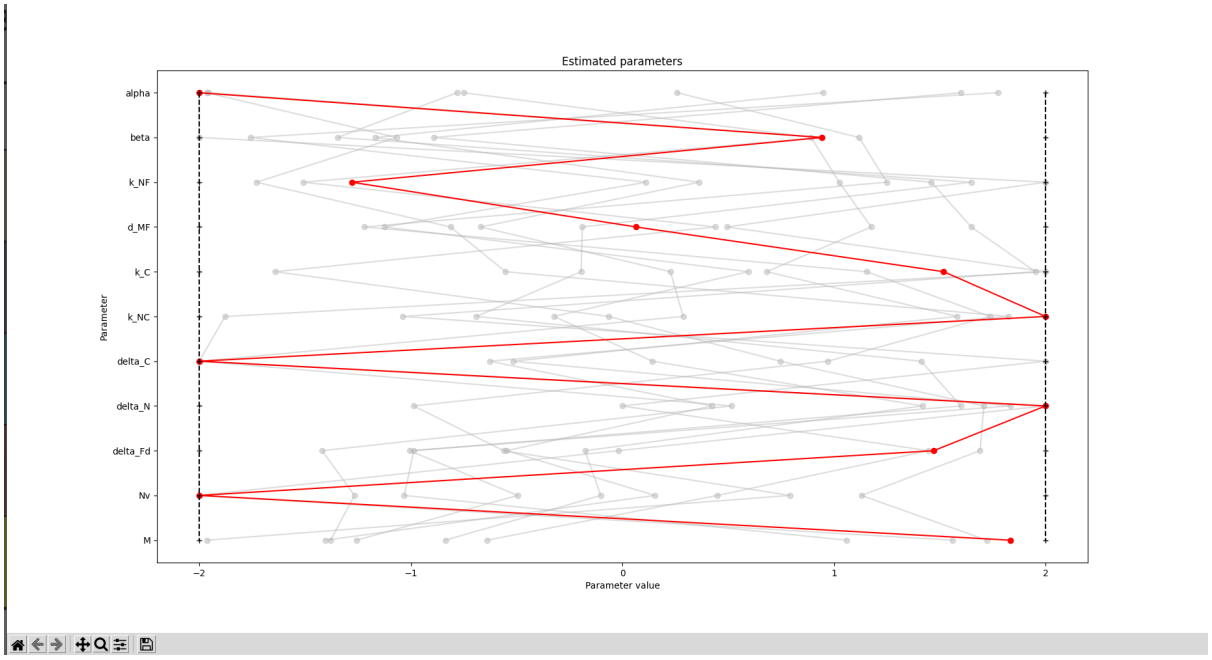


Figure 20 : Scipy default optimizer plot for parameters value: (max_iter=100)

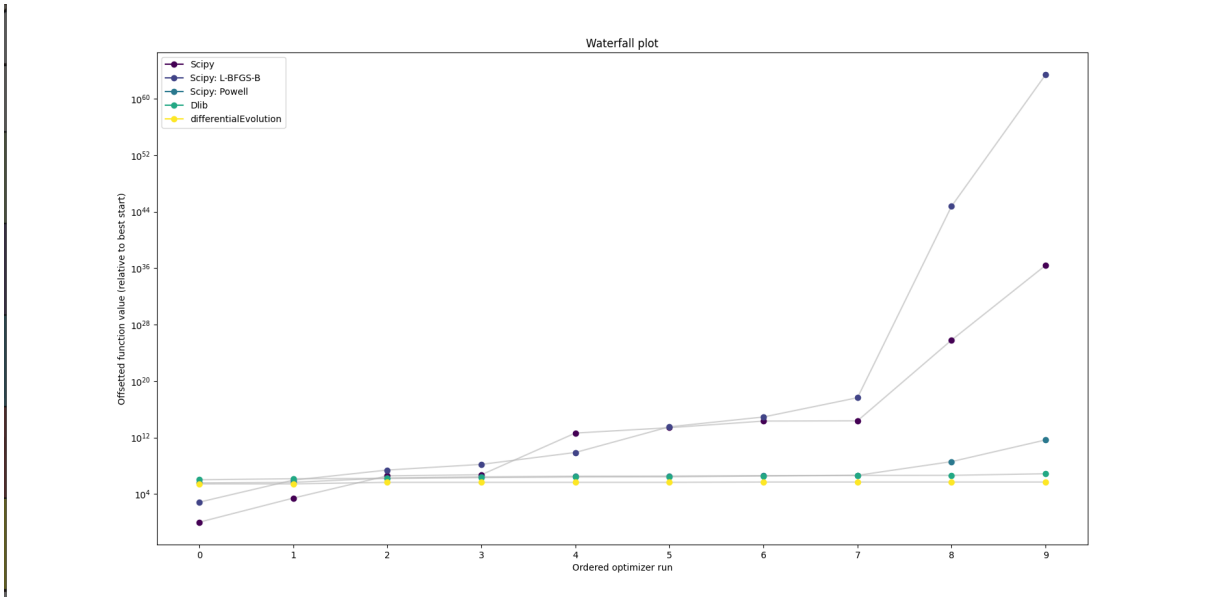


Figure 21 : Offsetted value of different optimizers

Average run time per start:

Scipy: 0.977018 s
Scipy: L-BFGS-B: 0.985179 s
Scipy: Powell: 37.038736 s
Dlib: 8.401999 s
pyswarm: 0.000347 s
differentialEvolution: 12.401379 s

Total time taken by the code to run=242.19592881202698 s

Average Run time per start:

Scipy: 0.761039 s
Scipy: L-BFGS-B: 0.869890 s
Scipy: Powell: 7.029892 s
Dlib: 0.256163 s
pyswarm: 0.000316 s
differentialEvolution: 10.623176 s
Total time taken by the code to run=135.77502703666687 s

Parameter uncertainties:

Result.sample_result['burn_in']
Geweke burn-in index: 0
Result.sample_result['effective_sample_size']
Estimated chain autocorrelation: 492.05292506440804

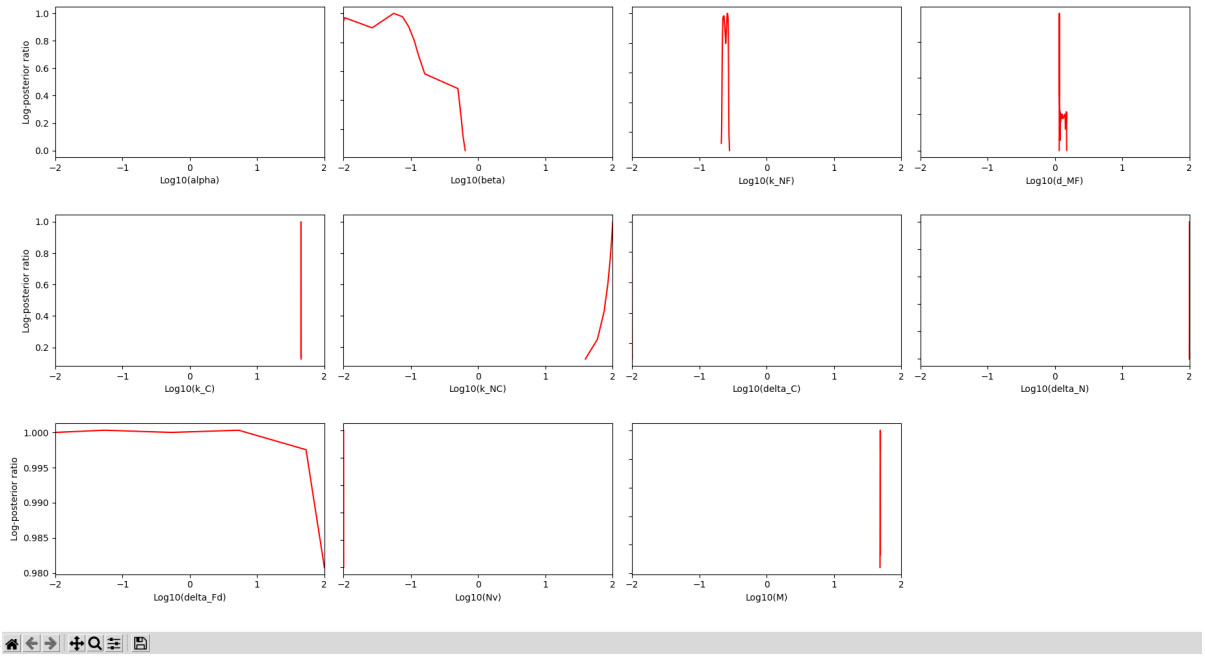


Figure 22 : Density vs log10(parameter) plot

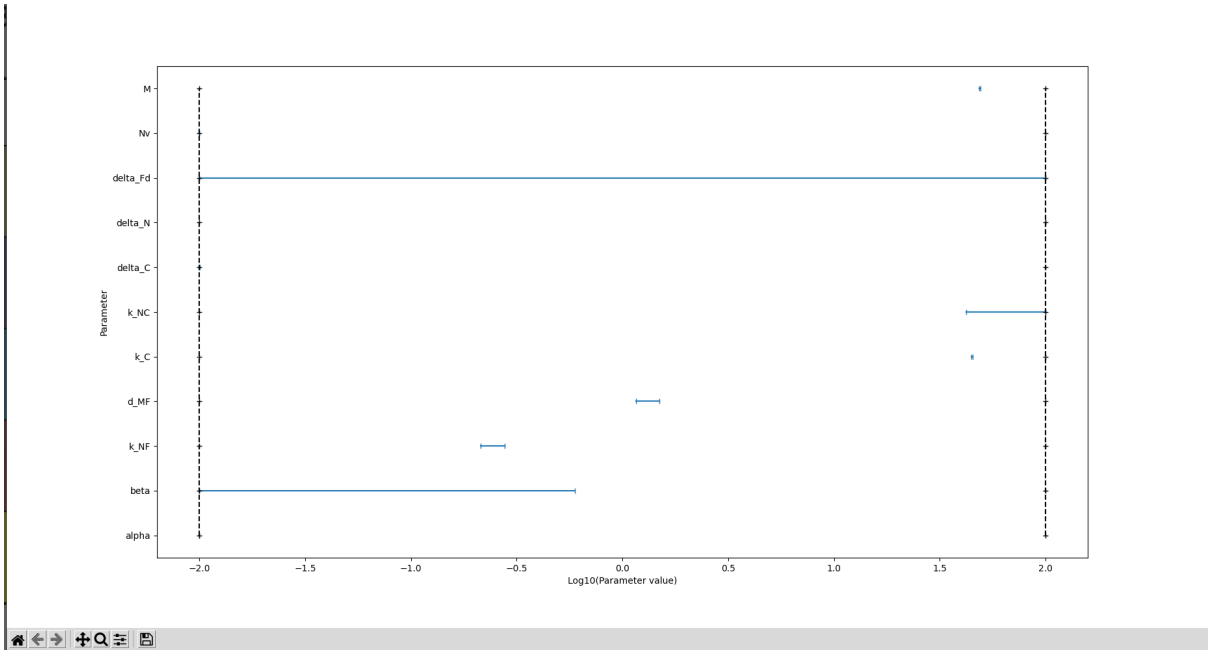


Figure 23 : Confidence intervals(range of values for parameters on a log10 scale)

Average Run time per start:

Scipy: 0.545509 s
Scipy: L-BFGS-B: 0.314934 s
Scipy: Powell: 11.474994 s
Dlib: 0.055811 s
pyswarm: 0.000313 s
differentialEvolution: 10.453979 s

Total time taken by the code to run = 434.3257668018341 s

For n_starts = 50

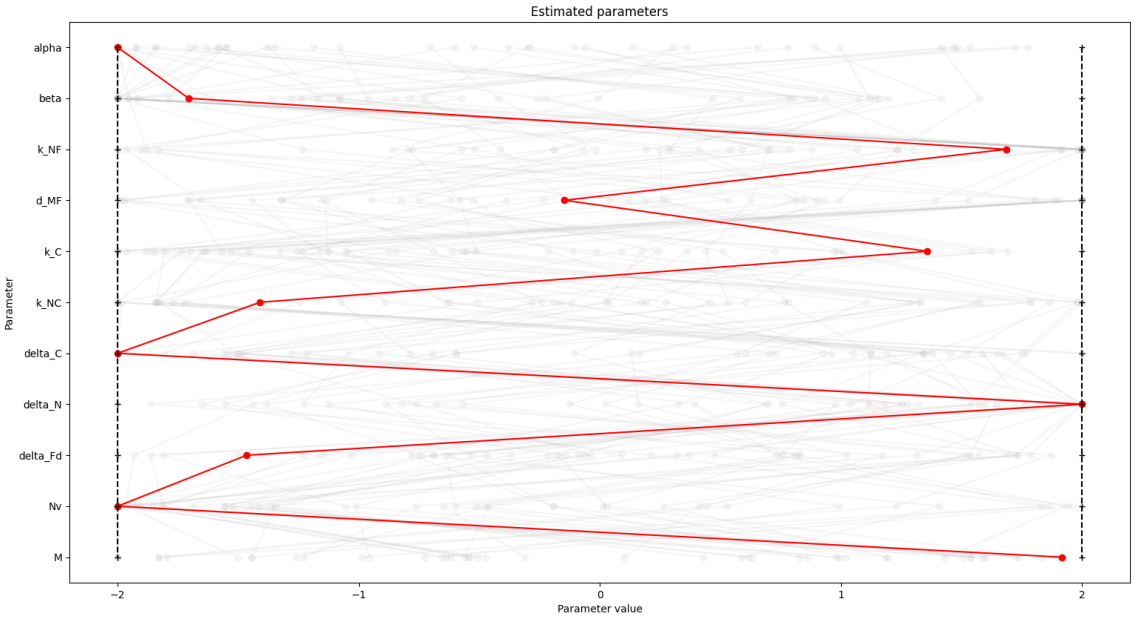


Figure 24 : Estimated parameter values for Scipy default optimizer

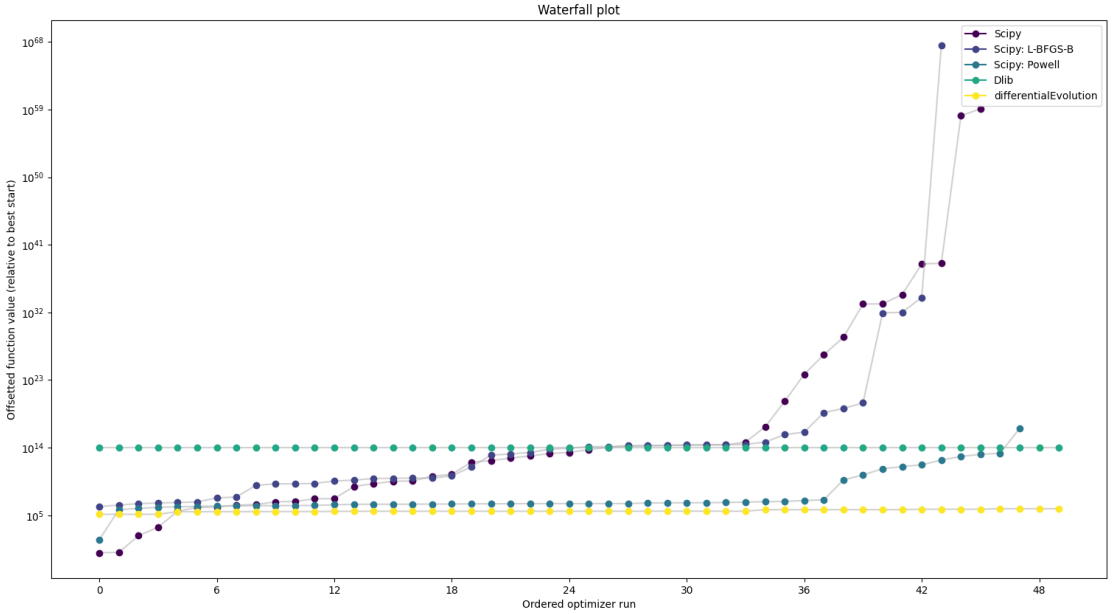


Figure 25 : Offsetted value of different optimizers

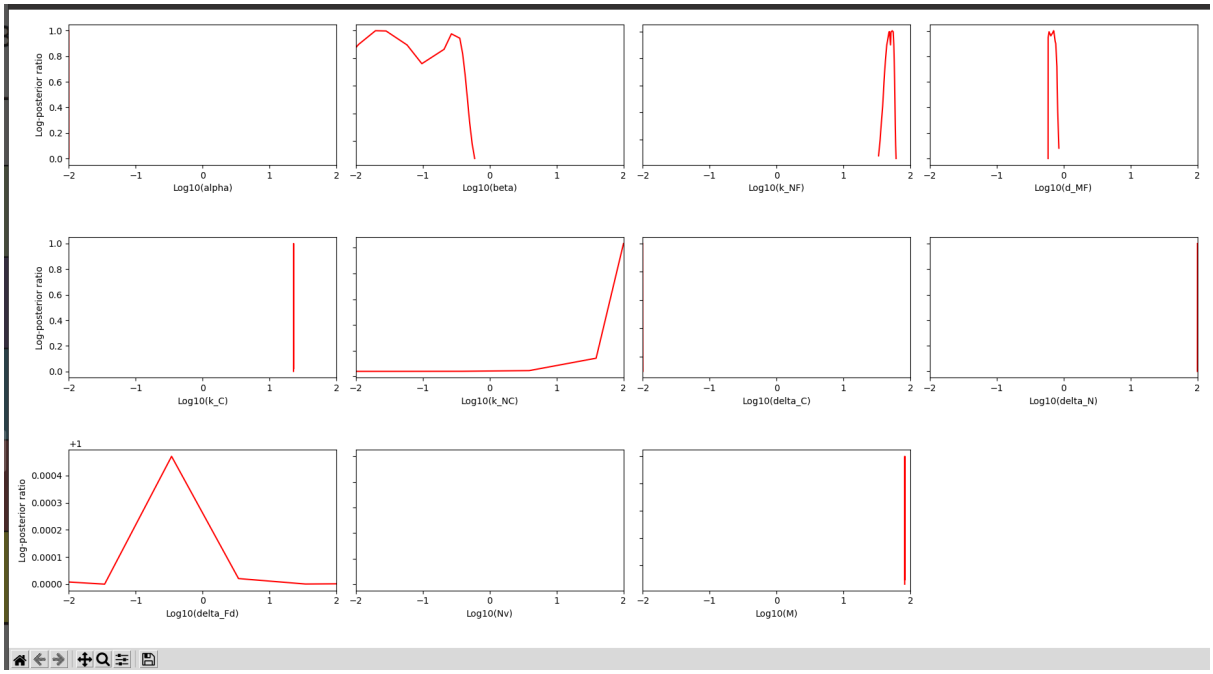


Figure 26 : Density vs log10(parameter) plot

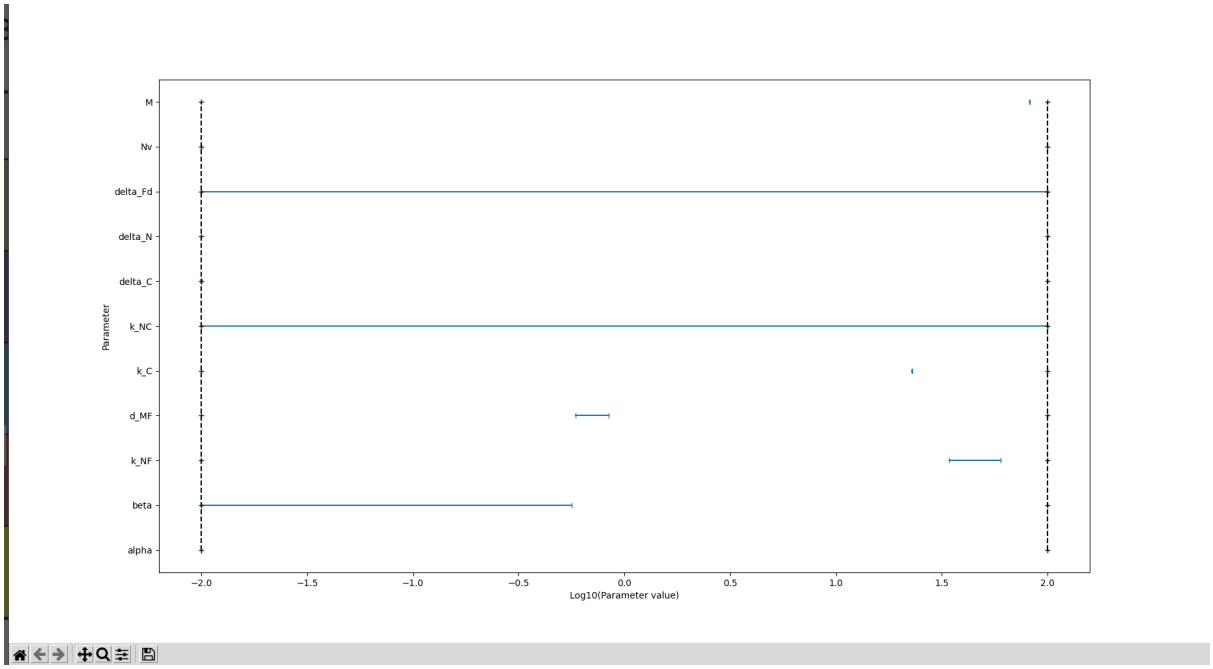


Figure 27 : Confidence intervals(range of values for parameters on a log10 scale)

Geweke burn-in index: 0
Estimated chain autocorrelation: 711.7839869891374

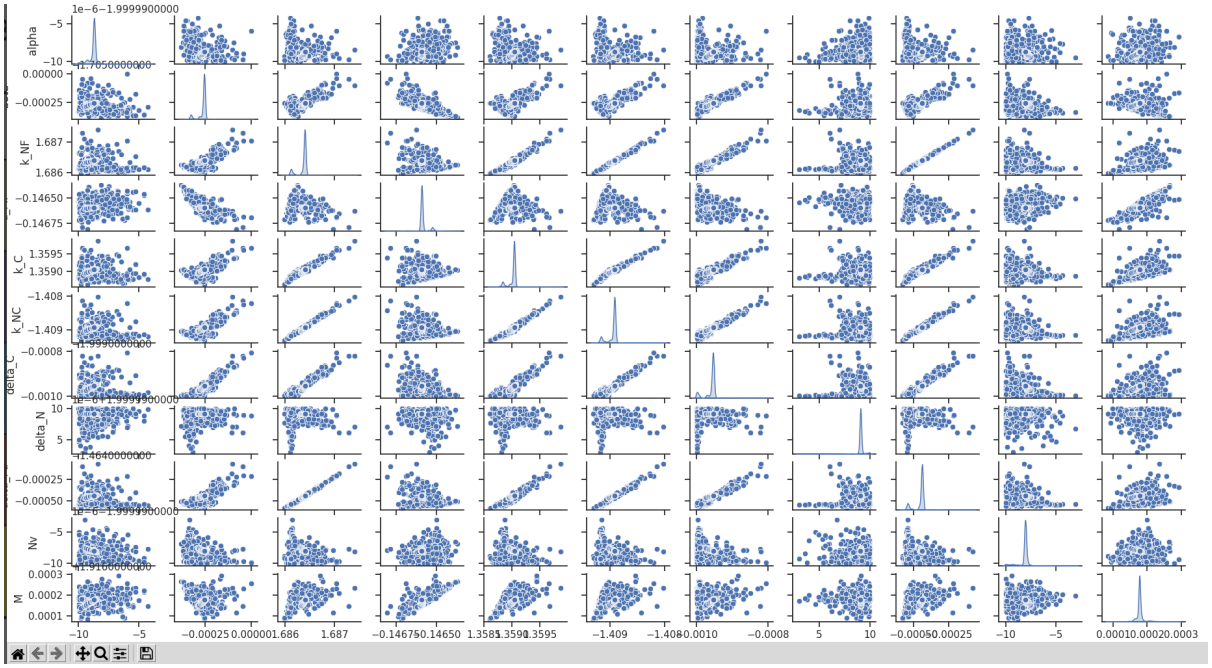


Figure 28 : Correlation plot of parameters

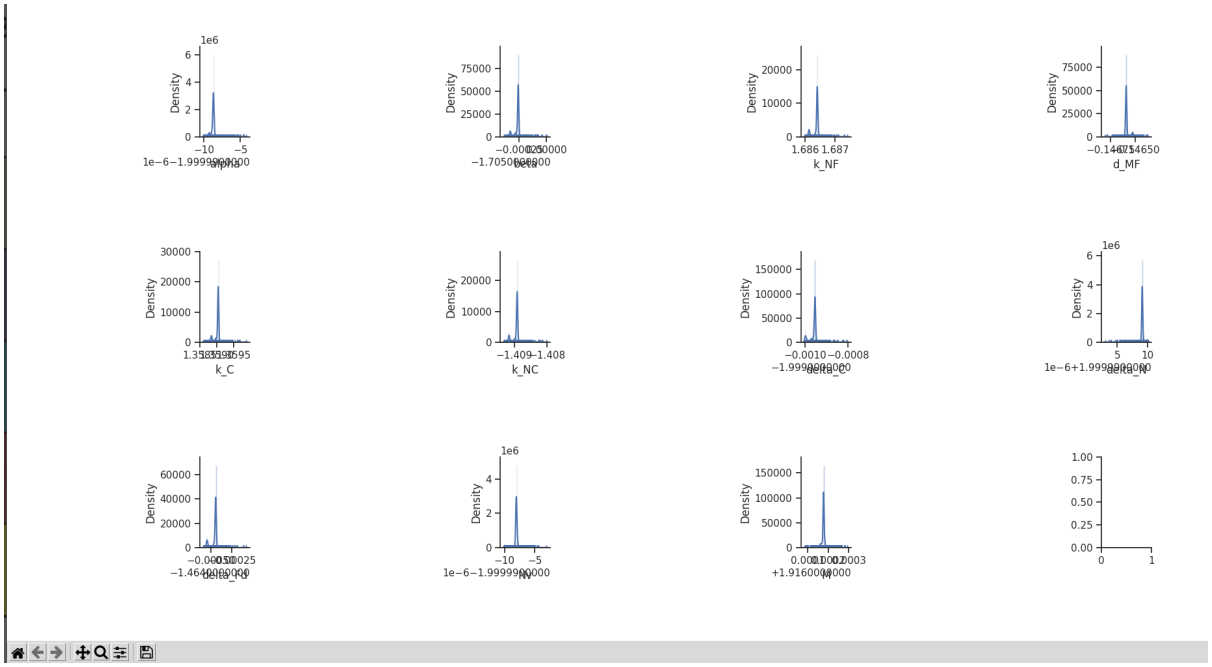


Figure 29 : Density vs Parameter value

Average Run time per start:

Scipy: 0.557576 s
Scipy: L-BFGS-B: 0.540394 s
Scipy: Powell: 13.146493 s
Dlib: 0.095311 s
pyswarm: 0.000383 s
differentialEvolution: 11.261650 s

Total time taken by the code to run = 608.5100643634796

- Codes and files:

All the files needed for the pypesto implementation are uploaded on Git. Here is the link

6. Final Results and Future Work

As the result of performing various optimization methods on Mechanistic Model in pypesto, following takeaways can be concluded:

- Parameter optimization:

For performing the parameter optimization task, extended yaml files to generate PEtab files and measurement file were used.

As a result of using pypesto optimization with scipy default optimizer, these values of the parameters were received.

alpha	-2
beta	-0.6411
k_NF	1.28052
d_Mf	-0.17508272,
k_C	1.32875188,
k_NC	-1.95231
delta_C	-2
delta_N	2
delta_Fd	1.79957
Nv	-2
M	1.95148

```yaml
parameters:
  - parameterId: alpha
    nominalValue: 0.0017
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: beta
    nominalValue: 0.28
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: k_NF
    nominalValue: 1.2e-6
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1
```

```yaml
  - parameterId: d_MF
    nominalValue: 0.32e-6
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: k_C
    nominalValue: 0.38e-12
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: k_NC
    nominalValue: 0.31e-6
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1
```

```yaml
  - parameterId: delta_C
    nominalValue: 0.066
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: delta_N
    nominalValue: 0.061
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: delta_Fd
    nominalValue: 0.1
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1
```

```
  - parameterId: Nv
    nominalValue: 1.5e+8
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1

  - parameterId: M
    nominalValue: 0.3e+6
    parameterScale: log10
    lowerBound: 0.01
    upperBound: 100
    estimate: 1
```

- Optimizer selection:

1. There were various bars to assess the performance of optimizers depending upon the constraint. The factors that contribute are n_starts and range of parameters.
2. Plots mainly to visualize which optimizer performs better than other optimizers. Waterfall plot can be best interpreted the offsetted function value vs number of fresh starts for all optimizers like this:

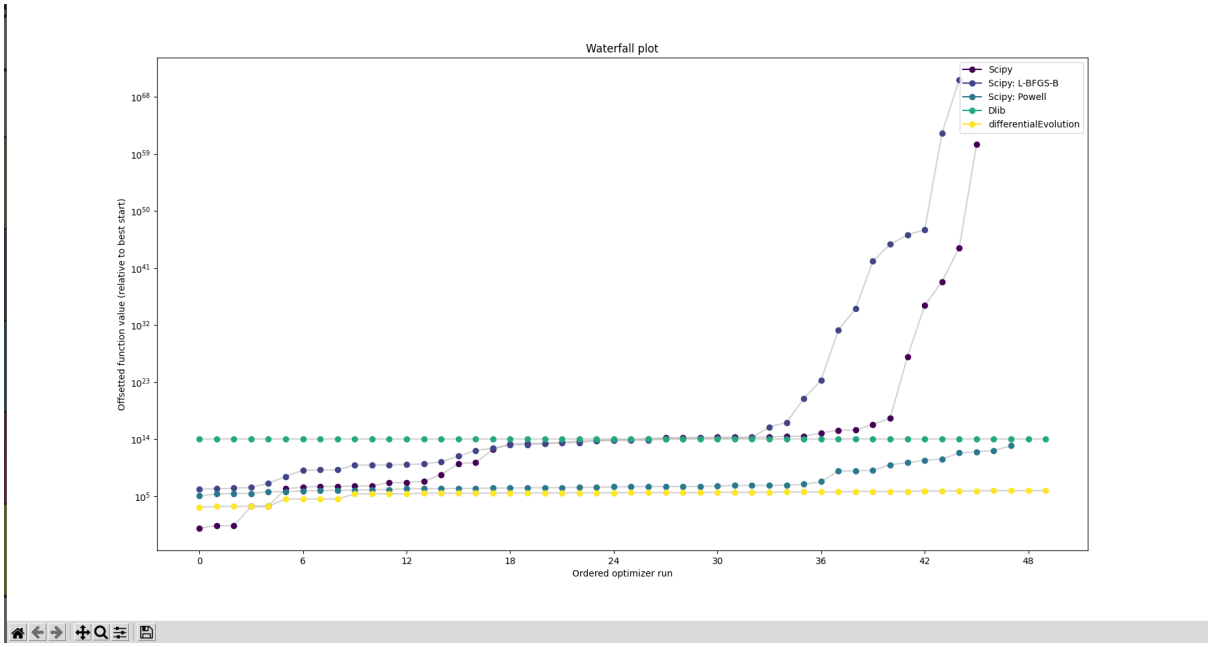


Figure 30 : Offsetted value of different optimizers

Time taken by the optimizers:

Average Run time per start:

Scipy: 0.557576 s
Scipy: L-BFGS-B: 0.540394 s
Scipy: Powell: 13.146493 s
Dlib: 0.095311 s
pyswarm: 0.000383 s
differentialEvolution: 11.261650 s

3. Plot to visualize the parameters value by the best optimizer can be seen like this:

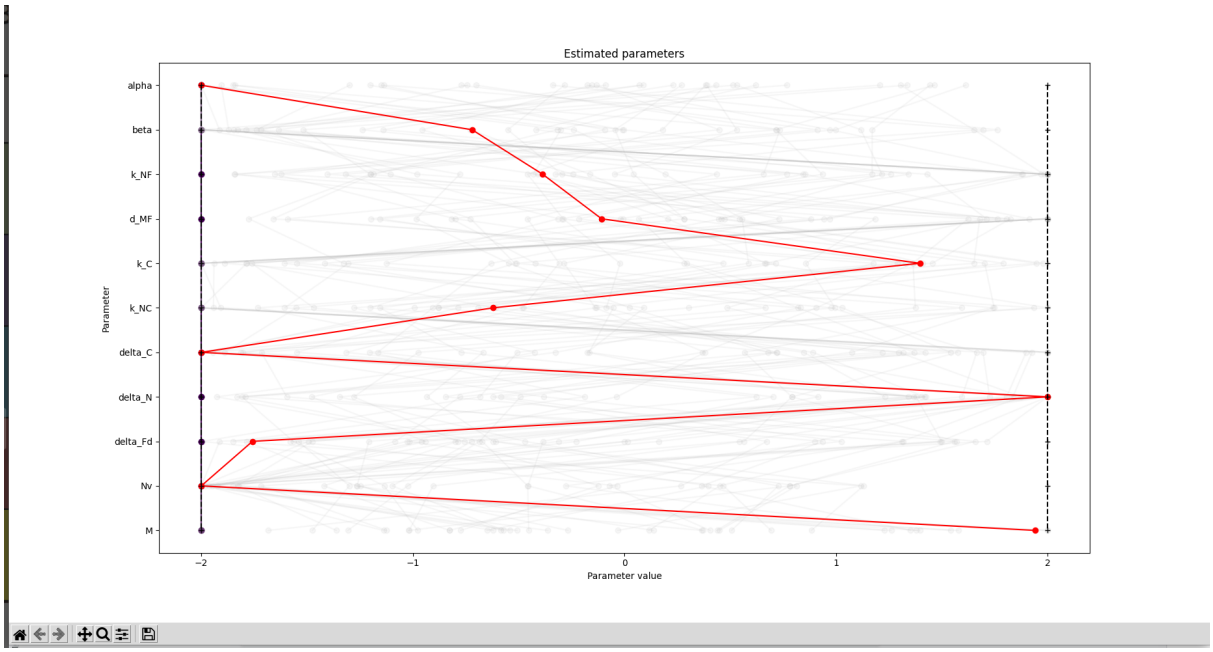


Figure 31 : Estimated parameter values for Scipy default optimizer

- In our case, the best optimizer was found to be the default scipy optimizer.
- In all the simulations done on optimizers, scipy optimizers performed better than others showing gradient based optimizers always perform better in systems biology dynamic systems.

4. Assessing the profile likelihood to verify if the best performing optimizer solution gives a peak within the lower and upper bound of parameters:

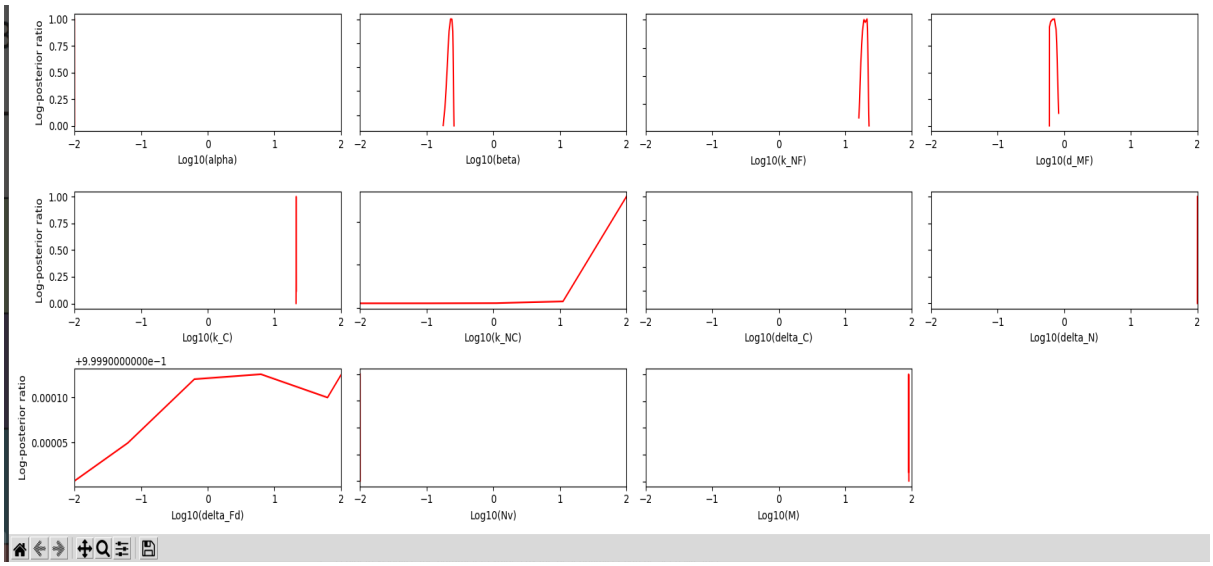


Figure 32 : Density vs log10(parameter) plot

- The visual interpretation helps identify the range in which the optimizer can find the best parameter value for a given max_iter and n_starts value.
- The confidence interval helps assess the range of values the parameter can take and how converged the solution is.

5. Sampling methods help plot different parameters density and correlation plots. Pypesto offers other sampling methods such as AdaptiveMetropolis, ParallelTempering, and Pymc3 that makes the whole process flexible and more to explore in pypesto.

Plots can be obtained by expanding the dataset with multiple conditions state, n_starts value, max_iters and other optimizers. Some of these variations can be seen in section 5.5.

7. Outlook (scope of current research work done in the project)

The following avenues could be explored as a next step:

1. Testing more optimization methods

- IpOpt (optimize.IpoptOptimizer())
    - Interior point method
- CMA-ES (optimize.CmaesOptimizer())
    - Covariance Matrix Adaptation Evolution Strategy
    - Evolutionary Algorithm

2. Tuning of n_starts and max_iters

As can be seen in the above results in the mechanistic model, smaller n_starts value does not help optimizers fit the data fastly and give optimized solutions. Increasing n_starts to a large number might result in huge time taken by the code to run. Similarly this is the case with max_iters. Hence one may try to optimize the n_starts value which balances optimized parameter fitting and time taken.

3. Increase the range of parameters to find optimized fit value

In some cases, we might see the blank part for a few parameters on plotting profile likelihood. This shows that no parameter value seemed to fit the dataset. Providing more broad range bounds to parameters could be a helpful idea to catch the optimized value density peak.
For eg. parameterId: alpha, the lowerbound and upperbound were lowerBound: 0.01 and upperBound: 100. It can be set to a lowerbound of 0.001 and upper bound of 1000.

8. Challenges and Summary

Challenges:

- Not widely used, a niche topic

This project content particularly required high-quality and authoritative on the studied topics. The successful optimization of various dynamic systems by applying pypesto has not been reported enough to date to the best of our knowledge. This might take more time in scaling up to become adequately advanced. Not much spreaded and yet to be used globally packages like amici, offered lots of hidden and

unexplored ways to open. Finally, sticking to a niche and writing in-depth about it was a real testament to establish credibility, understanding and absorption of untapped tools.

- Dig deeper to understand

One of the main challenges in applications of pypesto to solve parameter optimization problems stems from the intrinsic multidisciplinarity of the approach. Biochemists, molecular biologists, mathematicians, and computer scientists have a common language to clarify goals, carry out rigorous analysis and training, and avoid pitfalls, wrongful usage methods, and misinterpretations. This requires a deep understanding of the topic before using ready to use software packages. Trying to heap all the available data and running a range of optimizer algorithms to select the best predictor might not be the optimal strategy like a nonspecialist. The No Free Lunch theorem claims that no single method is superior to others a priori therefore, a thorough understanding of the data types to be used and problems to be solved is essential in developing efficient predictors.

- Get ready for lots of errors

With the continuous growth of parameter optimization applications in dynamic systems, the need for robust comparison of various predictors is of growing importance. This comparison is mainly obstructed by the lack of standardized protocols that results in a lot of errors before using software tools. The lack of insufficient disclosure in publications, and difficulty finding reviewers with sufficiently broad expertise are among the most pressing issues.

- Single progress involves lots of efforts

Finally, the excitement about parameter fitting application of pypesto to dynamic systems seems to put another critical component of the approach on the back burner. The ultimate goal of science is to achieve better predictive power and explain the results. To go beyond simple analysis and contribute in the currently evolving area the time taken to progress ratio is huge.

Summary:

Detailed procedures have been outlined to design parametric fitting models and the recommended/advised methods and descriptors that act as a practical reference for the subsequent studies. For optimization methods, it was found that the Scipy default optimizer model outperformed other optimizer models across different n_starts values. However, the ranking of the models depends on the given problem, data set, or model architecture. The rationale for the failure of the Differential Evolution and other non-gradient optimizers could be found in the fact that it is free from finding gradients(containing useful information) at any point. Gradients should be given weightage as our whole system is differential equation based (may be incorrect, to be specific it depends on the model).

9. References

[1].Tanaka, R. J. *et al. In silico* modeling of spore inhalation reveals fungal persistence following low dose exposure. *Sci. Rep.* **5**, 13958; doi: 10.1038/srep13958 (2015).

[2].Tara Hameed, “Invasive Aspergillosis Mechanistic Model”, June 4, 2021.

[3].Vanhoefer J., Matos, M. R. A., Pathirana, D., Schälte, Y. and Hasenauer, J. (2021). yaml2sbml: Human-readable and -writable specification of ODE models and their conversion to SBML. Journal of Open Source Software, 6(61), 3215, https://doi.org/10.21105/joss.03215

Advanced Multilanguage Interface to CVODES and IDAS, Link
[4].pyPESTO - A Python Package for Parameter Estimation and Uncertainty Quantification, https://pypesto.readthedocs.io/en/latest/contribute.html
[5].Benchmark-Models-PEtab, https://github.com/Benchmarking-Initiative/Benchmark-Models-PEtab
[6].Fröhlich, F., Weindl, D., Schälte, Y., Pathirana, D., Paszkowski, Ł., Lines, G.T., Stapor, P. and Hasenauer, J., 2021. AMICI: High-Performance Sensitivity Analysis for Large Ordinary Differential Equation Models. Bioinformatics, btab227, DOI:10.1093/bioinformatics/btab227.
[7]. Petr Dostál, “The Use of Optimization Methods in Business and Public Services” Handbook of Optimization, 2013, Volume 38 Link
[8].PyCharm-The Python IDE for Professional Developers, https://www.jetbrains.com/pycharm/
[9].Combine-2020-talk, https://github.com/jvanhoefer/COMBINE-2020-talk
[10].https://github.com/AMICI-dev/AMICI/issues/1535
[11].https://github.com/AMICI-dev/AMICI/issues/1533
[12].https://github.com/ICB-DCM/pyPESTO/issues/703
[13].https://github.com/ICB-DCM/pyPESTO/issues/701
[14].https://github.com/ICB-DCM/pyPESTO/issues/700
[15].https://github.com/ICB-DCM/pyPESTO/issues/698