



Matthias Frey :: PhD student :: Paul Scherrer Institut

## pyOPALTools – OPAL Pre- and Postprocessing Tools in Python

20 April 2020 :: Mini-Developer's Workshop

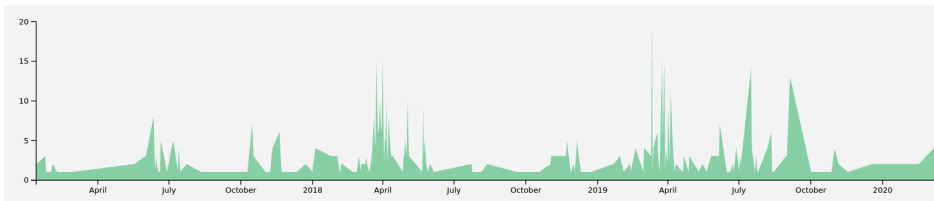


- data analysis
- machine learning
- uncertainty quantification
- batch job submission

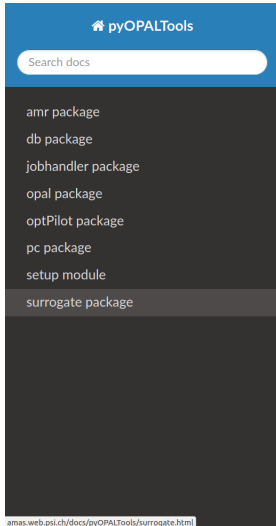
- Website: <https://gitlab.psi.ch/OPAL/pyOPALTools>
- Checkout: `git clone https://gitlab.psi.ch/OPAL/pyOPALTools.git`  
(developers: `git clone git@gitlab.psi.ch:OPAL/pyOPALTools.git`)
- Under development – contributors are welcome!

January 12, 2017 – April 11, 2020

Commits to master, excluding merge commits. Limited to 6,000 commits.



- Matthias Frey (PSI)
- Jochem Snuverink (PSI)
- Andreas Adelmann (PSI)
- Nicole Neveu (SLAC)
- Philippe Ganz (ETH Zürich)
- Renato Bellotti (ETH Zürich)
- Ryan Roussel (UCLA)



[Docs](#) » pyOPALTools Documentation

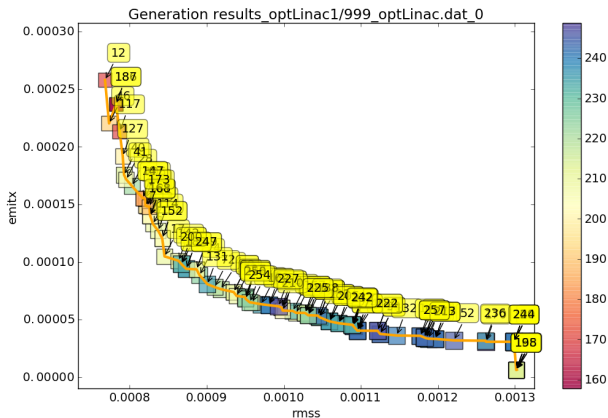
[View page source](#)

## pyOPALTools Documentation

### pyOPALTools

- [amr package](#)
  - [Submodules](#)
  - [amr.AmrOpal module](#)
- [db package](#)
  - [Submodules](#)
  - [db.mldb module](#)
- [jobhandler package](#)
  - [Submodules](#)
  - [jobhandler.submit module](#)
- [opal package](#)
  - [Subpackages](#)
    - [opal.analysis package](#)
      - [Submodules](#)
      - [opal.analysis.H5Statistics module](#)
      - [opal.analysis.OptimizerAnalysis module](#)

## optPilot Package - Pareto Front Example



(Taken from <https://gitlab.psi.ch/OPAL/opt-pilot/wikis/optpilot-week>)

- “mldb”
- Code written a while ago (2018-2019)
- Used for <https://journals.aps.org/prab/abstract/10.1103/PhysRevAccelBeams.23.044601>
- Could probably use some updating per user requirements.

Available functions:

```
['build', 'buildASCII', 'buildFromSDDS', 'buildFromSampler', 'getAllDvar',  
'getAllDvarData', 'getAllObj', 'getAllObjData', 'getBounds', 'getDVarVec',  
'getDvarsPD', 'getNumberOfSamples', 'getObjVec', 'getQoIsPD',  
'getQoiDvarPD', 'getSampleSize', 'getTimes', 'getXDim', 'getXNames',  
'getYDim', 'getYNames', 'load', 'printOverview', 'trainingSet', 'writeDB']
```

# Making a Database Example

Slide on courtesy of Nicole Neveu

```
from db import mldb

database = mldb.mldb()
database.build('dbtest', 'json-reader/data')
database.printOverview()
```

## Output:

```
OPAL ML Database Generator
Found 99 json files from 1 optimizers and 0 pareto front files
Write ML-Database dbtest.pk
xDim          = 4 -> ['benenergy', 'phiinit', 'prinit', 'rinit']
yDim          = 3 -> ['dpeak1', 'dpeak2', 'dpeak3_5']
generations   = 99
Data points   = 4741
Show first dataset from generation 0: y = f(x)
[43.37        7.17        9.33485] = f([ 7.10669e-02  1.07006e+02 -1.33397e-02
```



---

```
# import job submission class
from jobhandler.submit import JobSubmitter

# create job instance
js = JobSubmitter(sim_dirs = ['/my/first/simulation/directory/',
                              '/my/second/simulation/directory/'],
                  template = 'my/batch/file/input.slurm',
                  pair      = mydict,
                  cmd       = 'sbatch') # (e.g. SLURM)

# push to batch system
js.submit()
```

---

---

```
#!/bin/bash -l
#SBATCH --partition=@QUEUE@
#SBATCH --nodes=@NODES@
#SBATCH --time=@TIME@
#SBATCH --job-name=@SIM@
#SBATCH --error=@SIM@.error
#SBATCH --output=@SIM@.output
#SBATCH --clusters=merlin6

mpirun @EXEC@ @ARGS@
```

---

---

```
mydict = {
    'QUEUE': 'daily',
    'NODES': 4,
    'TIME': '4:00:00',
    'SIM': 'PSI-Ring',
    'EXEC': '$OPAL_EXE_PATH/opal',
    'ARGS': 'Ring.in'
}
```

---

```
import numpy as np

# set the domain of interest
pdom = np.empty((2, 1))
pdom[0, 0] = 0
pdom[1, 0] = 2.0 * np.pi

# load the UQ interface
from chaospy_model import UQChaospy

# setup the UQ analysis (lsq: least squares)
uq = UQChaospy(pdom, order=5, method='lsq')
```

---

---

<sup>1</sup><https://chaospy.readthedocs.io/en/master/>

<sup>2</sup><https://www.sandia.gov/UQToolkit/>

```
# create the training data
xtrain = np.random.uniform(pdom[0, 0], pdom[1, 0], (50, 1))
ytrain = np.sin(xtrain)

# train the surrogate model
uq.fit(xtrain, ytrain)

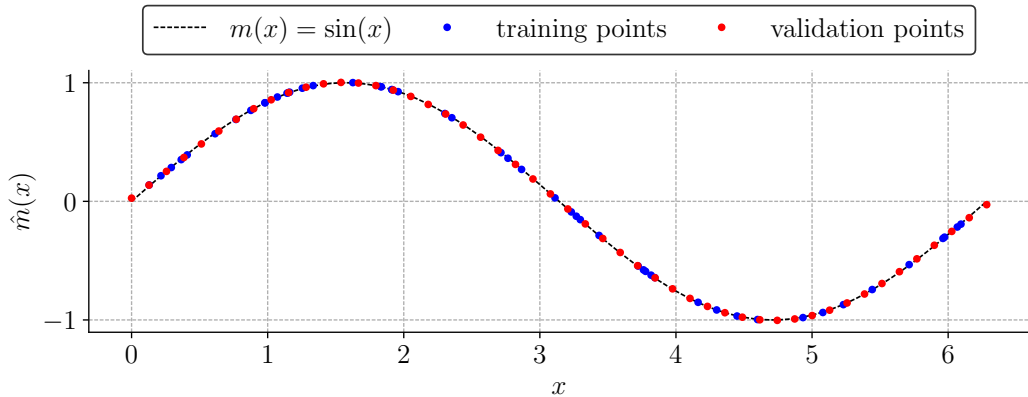
# predict the model response
ypred = uq.predict(xtrain)

# 95% confidence interval of the Sobol' indices
# std_main: standard deviation of the main sensitivities
# std_total: standard deviation of the total sensitivities
from bootstrap.bootstrap import bootstrap_sobol
std_main, std_total = bootstrap_sobol(xtrain, ytrain, uq,
                                     n_boot=100, seed=42)
```

---

# Surrogate Package – UQ with chaospy or UQTk

Work in Progress



## Data Loading

---

```
from opal import load_dataset, filetype

# load a specific file (here HDF5) using its file name
ds = load_dataset(directory='/path/to/files/',
                  fname='filename.h5')

# show summary of dataset
print(ds)

# load files using their file type (here HDF5)
dsets = load_dataset(directory='/path/to/files/',
                    ftype=filetype.H5)
```

---

## Data Access

---

```
# obtain array of positions of macroparticles in x-direction at step 1
x_data  = ds.getData(var='x', step=1)

# unit of positions
x_unit  = ds.getUnit(var='x')

# variable name, mainly used as axis labels
x_label = ds.getLabel(var='x')
```

---

file type	format	description
AMR	HyperCLaw-V1.1 <sup>3</sup>	particle phase space and AMR grid data
GRID	SDDS	AMR load balancing
HIST	ASCII	histogram of the radial positions of the particles at probe elements
H5	HDF5	particle phase space, used as a check point file
LBAL	SDDS	load balancing of macroparticles
LOSS	ASCII	phase space data of particles lost at elements (e.g., collimator, stripper)
MEM	SDDS	virtual memory consumption
OPTIMIZER	JSON	generation files and Pareto file of multiobjective genetic algorithm
OUTPUT	ASCII	OPAL standard output
PEAK	ASCII	peak values of the histogram generated by a radial probe
SAMPLER	JSON	all individual simulations of a sample run
SMB	SDDS	statistical measures of individual bunches of a neighboring bunch model
SOLVER	SDDS	AMR multigrid solver convergence information
STAT	SDDS	statistical measures of the bunch
TIMING	ASCII	runtime of individual code sections
TRACK_ORBIT	ASCII	single particle phase space data

<sup>3</sup>A file format of AMReX.



## Data Plotting

---

```
# set the plotting style
from opal.visualization.styles import load_style

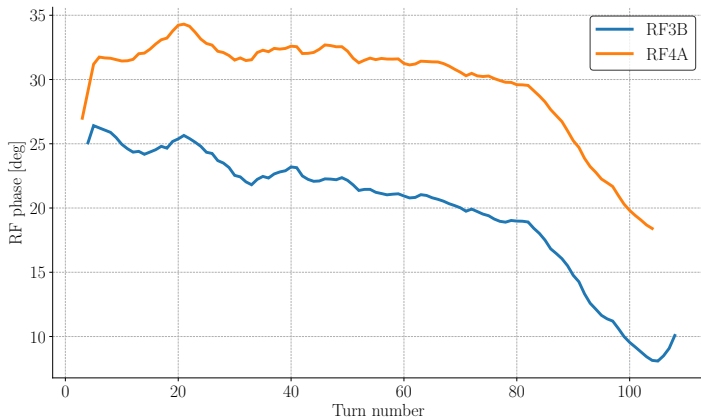
load_style('jupyter') # 'default' or 'poster'

# load the dataset
from opal.opal import load_dataset
ds = load_dataset('.././../tests/testData/', fname='Accelerated.out')

# create RF phase plot
plt = ds.plot_RF_phases(RFcavity=['RF3B', 'RF4A'])

# modify plot
plt.grid(linestyle='dashed')
plt.show()
```

# OPAL Package – Unified interface to do data analysis



## Data Plotting

---

```
ds = load_dataset('./RingCyclotron/', fname='RingCyclotron.stat')

plt.figure(figsize=(9, 4))
plt = ds.plot_profile1D('time', 'rms_x')

plt.grid(linestyle='dashed')

plt.tight_layout()

plt.show()
```

---

