PAUL SCHERRER INSTITUT

PSI

Matthias Frey :: PhD student :: Paul Scherrer Institut

pyOPALTools – OPAL Pre- and Postprocessing Tools in Python

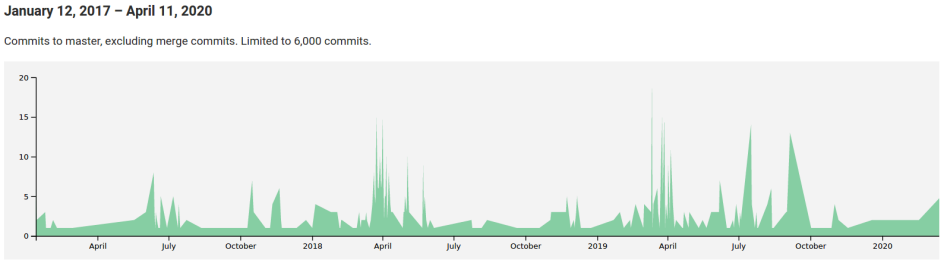
20 April 2020 :: Mini-Developer's Workshop

pyOPALTools provides tools for ...

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

Repository on PSI GitLab

- 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!

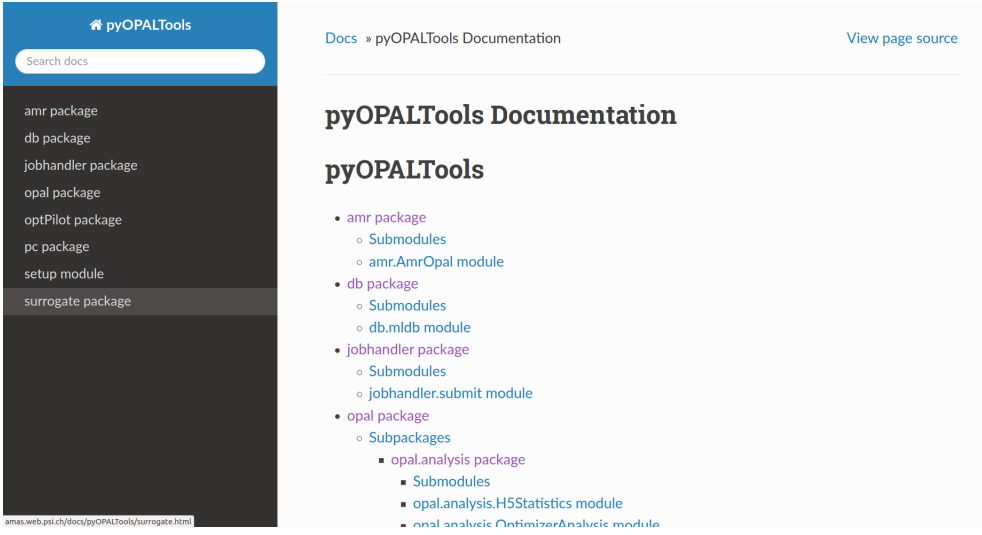


PAUL SCHERRER INSTITUT PSI

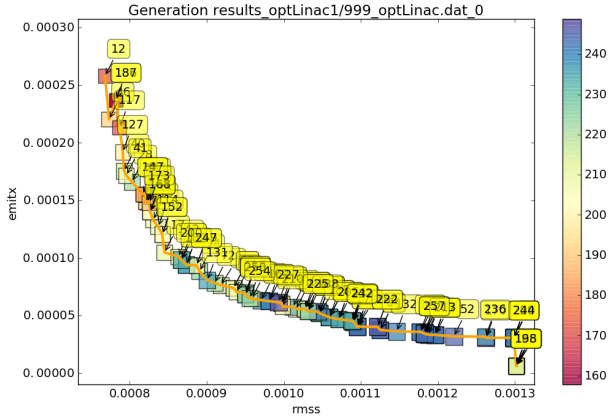
Contributions by

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

Sphinx Documentation – Link via Wiki of Repository



optPilot Package - Pareto Front Example



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

Job Submission Package

```
# 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()
```

Template Input Example

```
#!/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'
}
```

Surrogate Package – UQ with chaospy[1] or UQTk[2]

Work in Progress

```python
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')
```

[1] https://chaospy.readthedocs.io/en/master/

[2] https://www.sandia.gov/UQToolkit/

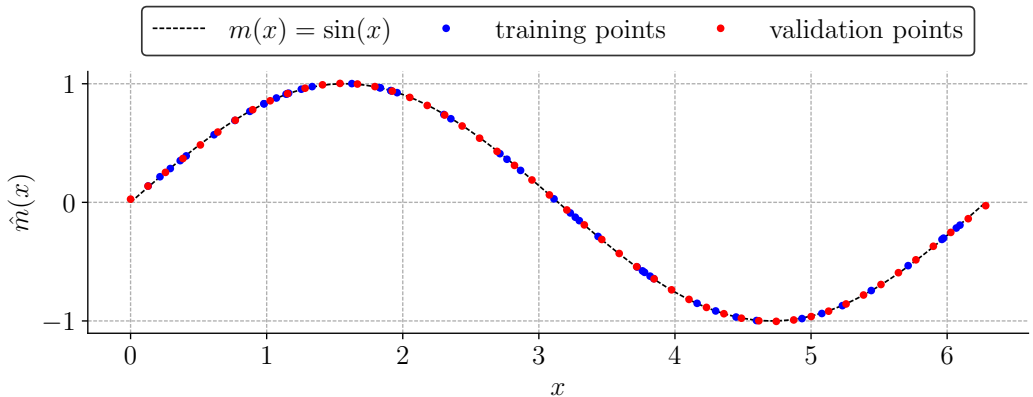
Surrogate Package – UQ with chaospy or UQTk

Work in Progress

```
# 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



OPAL Package – Unified interface to do data analysis

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)
```

OPAL Package – Unified interface to do data analysis

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')
```

OPAL Package – Unified interface to do data analysis

file type	format	description
AMR	HyperCLaw-V1.1[3]	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

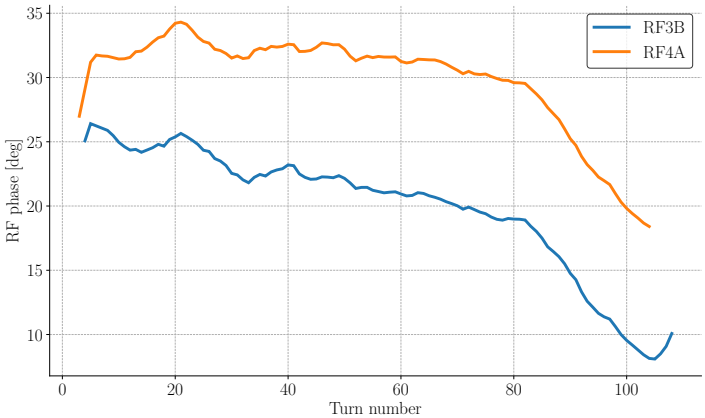
[3] A file format of AMReX.

OPAL Package – Unified interface to do data analysis

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



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()
```

OPAL Package – Unified interface to do data analysis