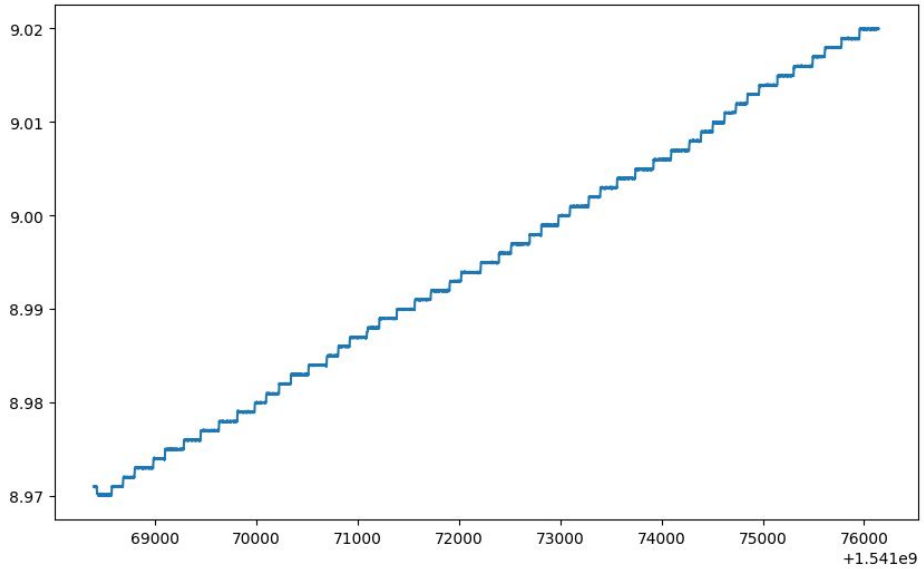
High-level analysis building blocks

Philipp Schmidt
DA

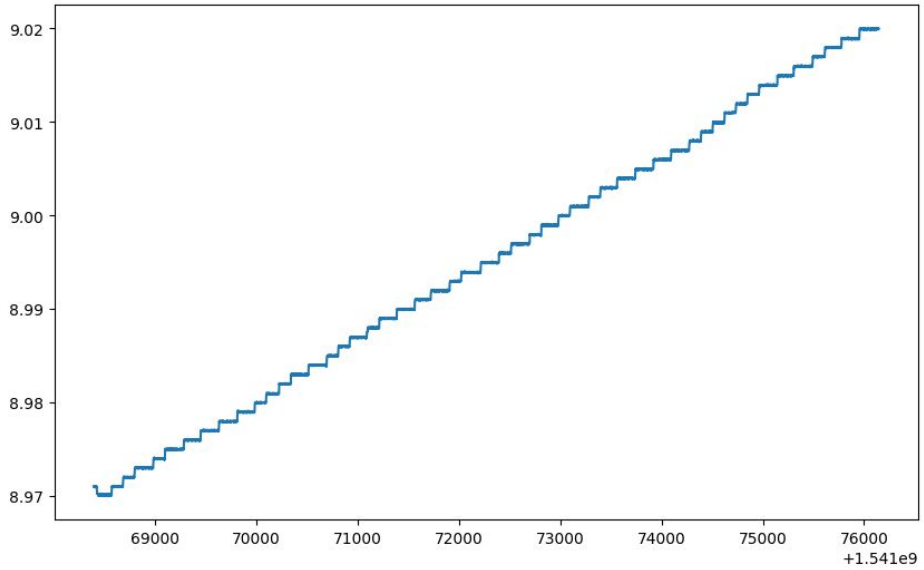
Hamburg

Accessing experimental data at European XFEL




```
run = extra_data.open_run(3495, 149)
plt.plot(run['…/MOTOR/…', 'actualPosition'].series())
```

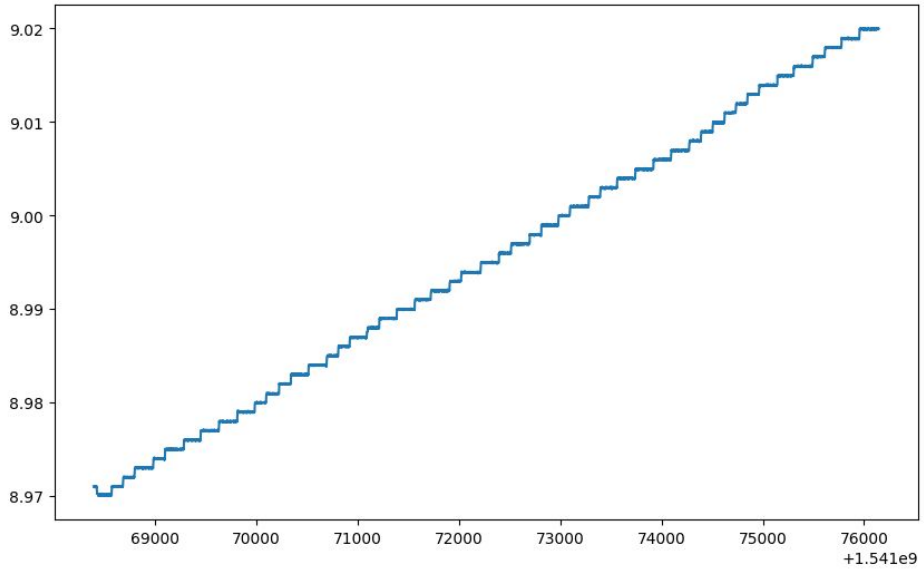
Accessing experimental data at European XFEL




```
run = extra_data.open_run(3495, 149)
plt.plot(run['…/MOTOR/…', 'actualPosition'].series())
```

```
motor = '…/MOTOR/…'
aggregator = 'DA01'
run_path = '/gpfs/exfel/exp/…/r0149'

train_ids = []
mono_pos = []
for h5path in Path(run_path).glob(f'*-{aggregator}-*.h5'):
    with h5py.File(h5path, 'r') as h5file:
        tids = h5file['INDEX/trainId']
        first = h5file[f'INDEX/{motor}/first'][np.argsort(tids)]
        vals = h5file[f'CONTROL/{motor}/actualPosition/value'][first]

        train_ids.append(sorted(tids))
        mono_pos.append(vals)

train_ids = np.concatenate(train_ids)
mono_pos = np.concatenate(mono_pos)

plt.plot(train_ids, mono_pos)
```

Accessing experimental data at European XFEL



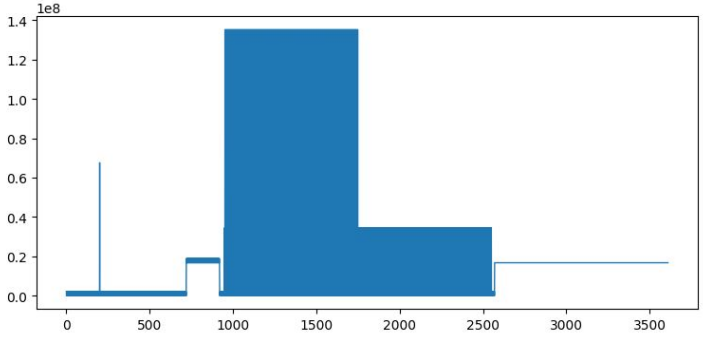

```
run = extra_data.open_run(3495, 149)
plt.plot(run.alias['photon-energy'].series())
```

```
motor = '…/MOTOR/…'
aggregator = 'DA01'
run_path = '/gpfs/exfel/exp/…/r0149'

train_ids = []
mono_pos = []
for h5path in Path(run_path).glob(f'*-{aggregator}-*.h5'):
    with h5py.File(h5path, 'r') as h5file:
        tids = h5file['INDEX/trainId']
        first = h5file[f'INDEX/{motor}/first'][np.argsort(tids)]
        vals = h5file[f'CONTROL/{motor}/actualPosition/value'][first]

        train_ids.append(sorted(tids))
        mono_pos.append(vals)

train_ids = np.concatenate(train_ids)
mono_pos = np.concatenate(mono_pos)

plt.plot(train_ids, mono_pos)
```

Accessing experimental data at European XFEL

```
pulses = run.alias['pulse_table'].ndarray()
plt.plot(pulses[0])
pulses
```

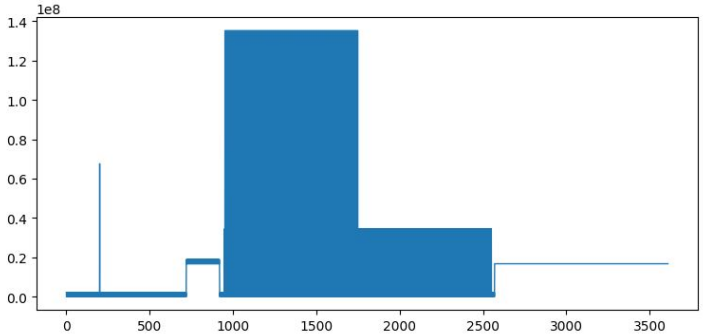
```
array([[ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       ...,
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216]],
      dtype=uint32)
```



Accessing experimental data at European XFEL

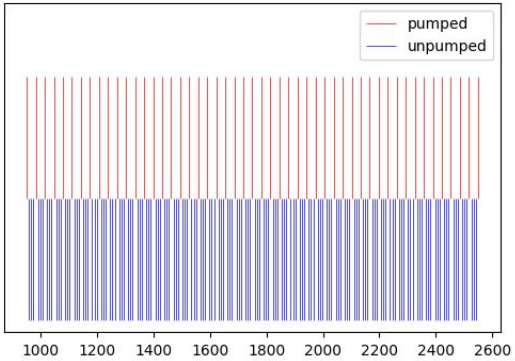
```
pulses = run.alias['pulse_table'].ndarray()
plt.plot(pulses[0])
pulses
```

```
array([[ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       ...,
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216],
       [ 2113385,        0,  2097257, ..., 16777216, 16777216, 16777216]],
      dtype=uint32)
```

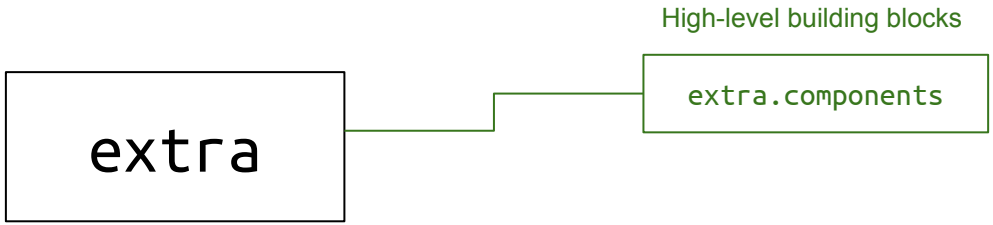



```
pulses = PumpProbePulses(run)
pids = pulses.peek_pulse_ids()

plt.vlines(pids[:, True, True], 0, 1, 'red', label='pumped')
plt.vlines(pids[:, True, False], 0, -1, 'blue', label='unpumped')
pids
```

```
pulseIndex  fel   ppl
0           True  True      950
1           True  False     958
2           True  False     966
3           True  False     974
4           True  True      982
                           ...
196         True  True     2518
197         True  False    2526
198         True  False    2534
199         True  False    2542
200         True  True     2550
Length: 201, dtype: int32
```

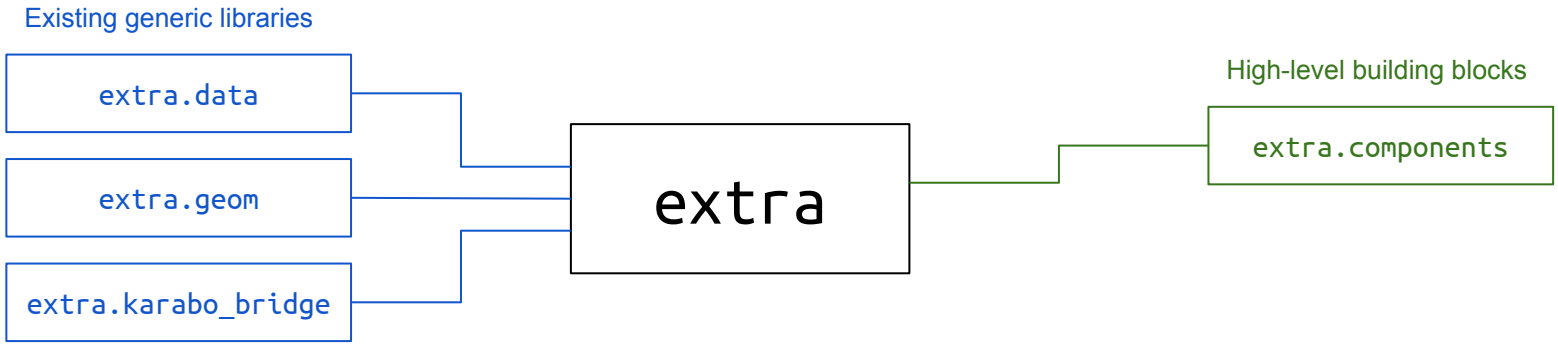


High-level analysis building blocks



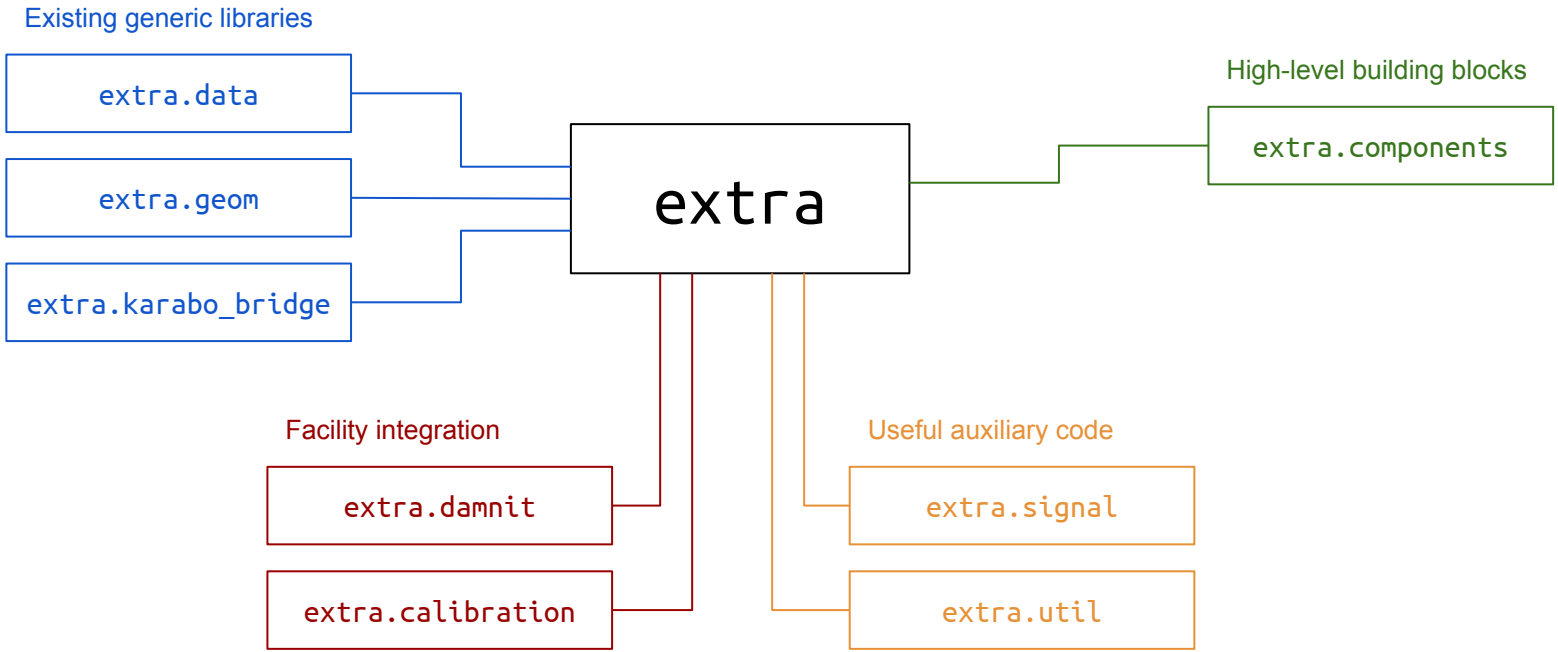
EXtra = European XFEL Toolkit
for Research and Analysis

High-level analysis building blocks



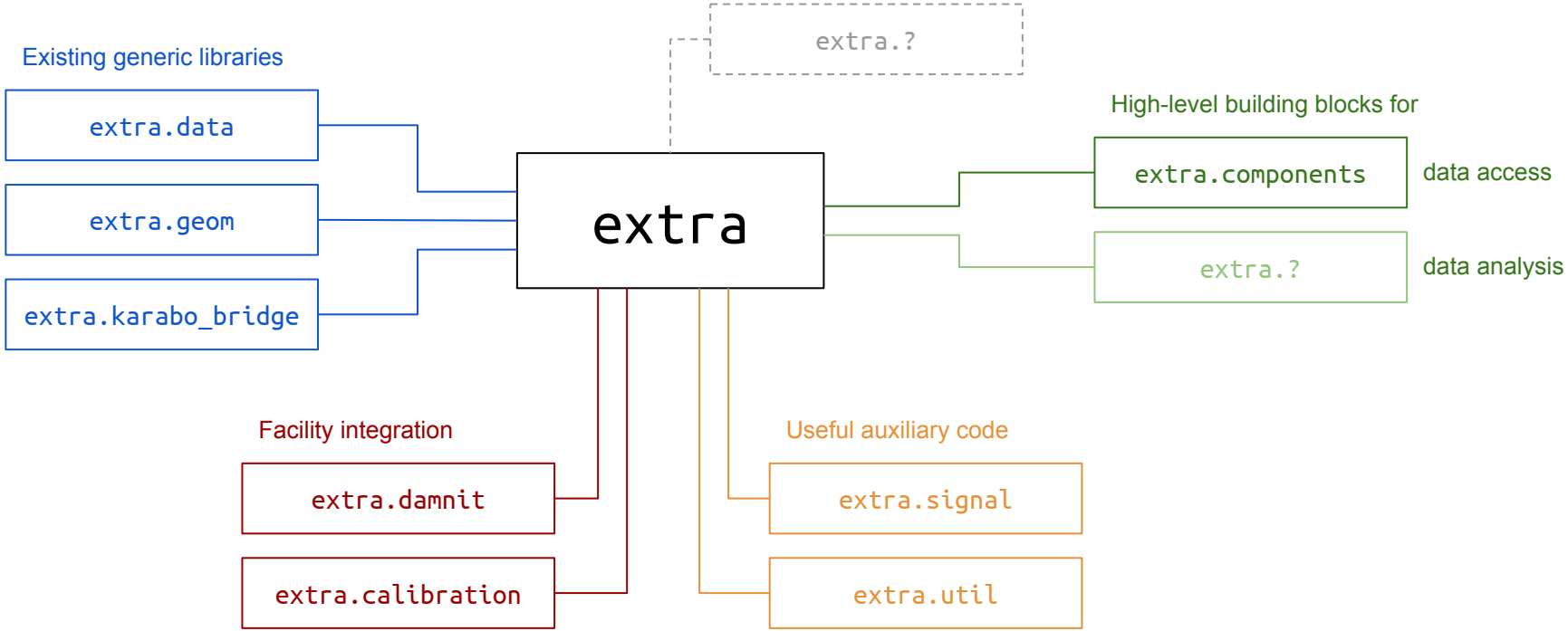
EXtra = European XFEL Toolkit
for Research and Analysis

High-level analysis building blocks



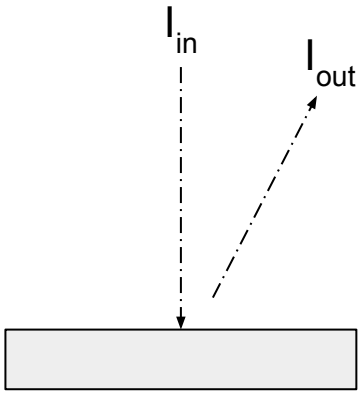
EXtra = European XFEL Toolkit
for Research and Analysis

High-level analysis building blocks



EXtra = European XFEL Toolkit
for Research and Analysis

Hands-on analysis

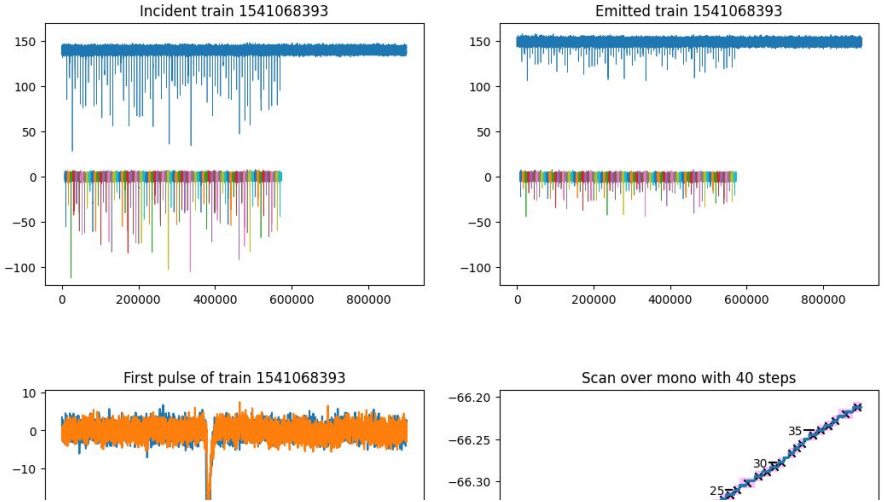


jupyter extra-components-UM2025 Last Checkpoint: 1 hour ago

File Edit View Run Kernel Git Settings Help Trusted

Code JupyterLab EXtra

```
in_train0_pulses = in_pulses.sel(trainId=run.train_ids[0])
in_train0_full = in_channel.raw_samples_key[0].ndarray()[0]

inp.plot(pulse_coords(in_train0_pulses).T, in_train0_pulses.T, lw=0.5)
inp.plot(in_train0_full + 150, lw=0.5)

out_train0_pulses = out_pulses.sel(trainId=run.train_ids[0])
out_train0_full = out_channel.raw_samples_key[0].ndarray()[0]

outp.plot(pulse_coords(out_train0_pulses).T, out_train0_pulses.T, lw=0.5)
outp.plot(out_train0_full + 150, lw=0.5)

pulsep.plot(in_pulses.isel(pulse=0), label='Incident')
pulsep.plot(out_pulses.isel(pulse=0), label='Emitted')
pulsep.legend()

scan.plot(ax=scanp);
```

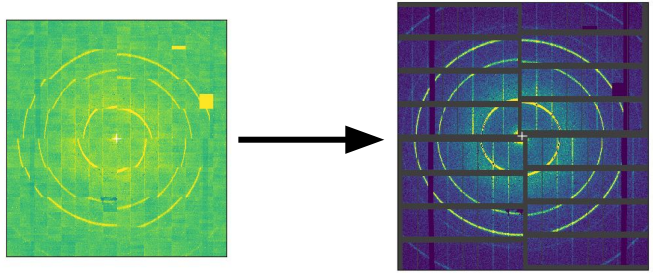


EXtra components

- Growing list of components
  - `AdqRawChannel`
  - `AGIPD1M`
  - `AGIPD1MQuadrantMotors`
  - `AGIPD500K`
  - `DelayLineDetector`
  - `DldPulses`
  - `DSSC1M`
  - `JUNGFRAU`
  - `MachinePulses`
  - `LPD1M`
  - `Scan`
  - `Scantool`
  - `OpticalLaserDelay`
  - `OpticalLaserPulses`
  - `PumpProbePulses`
  - `Timepix3`
  - `XGM`
  - `XrayPulses`
- Something missing you need? Please contact us at da@xfel.eu or on GitHub.

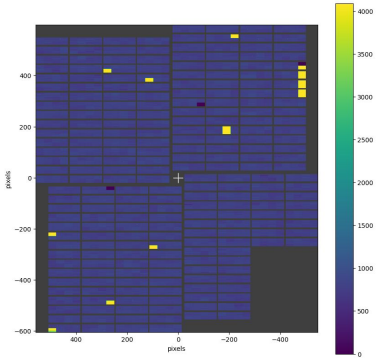
Integrate with facility infrastructure: `CalibrationData`

- Access the same calibration data as EuXFEL uses to provide corrected data, e.g. detector pedestals, gain characterization curves and more



```
from extra.calibration import CalibrationData, LPDConditions

lpd_cond = LPDConditions(memory_cells=200, sensor_bias_voltage=250)
lpd_cd = CalibrationData.from_condition(
    lpd_cond, 'FXE_DET_LPD1M-1', event_at='2022-05-22T02:00:00')

lpd_cd.display_markdown_table()

from extra.geom import LPD_1MGeometry
offsets = lpd_cd['Offset'].xarray()
LPD_1MGeometry.example().plot_data(offsets[:, :, :, 40, 2])
```

Modules	BadPixelsDark	Noise	Offset	ThresholdsDark
0	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34
1	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34
2	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34
...				
14	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34
15	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34	2023-07-29 19:34



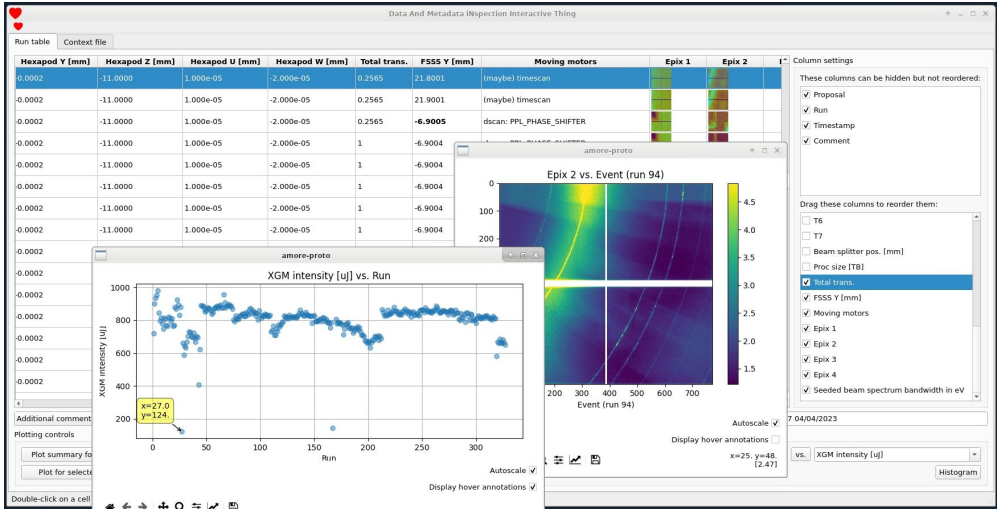
Integrate with facility infrastructure: DAMNIT

Visit our poster tomorrow: *Automatic experiment overview with DAMNIT*

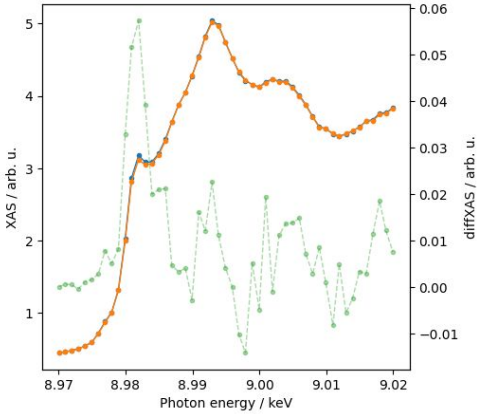
- Access the data saved in your proposal's DAMNIT database for further exploration or analysis

```python
from extra.damnit import Damnit

db = Damnit(6996)

xpcs = db[32, 'xpcs_mean_dataset']
db.table(with_titles=True)
```

	Proposal	Run	Timestamp	Total trans.	Opt. trans.	Sample Y	XPCS kbar	comment	XPCS PIPELINE	XPCS q-rings	...	Sample velocity	XPCS TTCF's	Rep rate [MHz]	Samp
0	6996	1	2024-05-30 06:01:03.121606112+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	
1	6996	2	2024-05-30 06:01:39.430480003+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	
2	6996	3	2024-05-30 06:02:17.112767935+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	
3	6996	4	2024-05-30 10:30:59.461067915+02:00	0.000115	0.006811	21.197261	None	None	None	None	...	2.0	None	2.256944	-1.234
4	6996	5	2024-05-30 10:33:38.049537897+02:00	0.000115	0.006811	21.225508	None	None	None	None	...	0.4	None	2.256944	-20.883
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
422	6996	425	2024-06-02 06:43:57.187414885+02:00	0.000060	0.002750	-3.181907	<image>	None	None	<image>	...	0.4	<image>	2.256944	-0.242
423	6996	426	2024-06-02 06:52:12.155587912+02:00	0.000259	0.011879	18.184154	<image>	None	None	<image>	...	0.4	<image>	2.256944	-0.240
424	6996	427	2024-06-02 07:00:32.948685884+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	
425	6996	428	2024-06-02 07:01:40.818312883+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	
426	6996	429	2024-06-02 07:02:48.240273952+02:00	NaN	NaN	NaN	None	None	None	None	...	NaN	None	NaN	



EXtra is…

- A single entry point for general purpose libraries like EXtra-data or EXtra-geom
- Integrate with facility infrastructure like calibration data or DAMNIT databases
- Components as high-level building blocks for data access
- *Coming soon: Experiment-specific analysis building blocks*
- Find out more at https://extra.readthedocs.io or contact us via da@xfel.eu

```
run = open_run(3495, 149)
mono = Scan(run.alias['photon_energy'])
pulses = PumpProbePulses(run, pulse_offset=1)

in_pumped = AdqRawChannel(run, '2_B', pulses=pulses).pulse_data() \
    .sel(fel=True, ppl=True, sample=np.s_[2920:3150])
out_pumped = AdqRawChannel(run, '2_A', pulses=pulses).pulse_data() \
    .sel(fel=True, ppl=True, sample=np.s_[2920:3150])

pumped_scan = scan.bin_by_steps(
    out_pumped.groupby('trainId').mean(...) / in_pumped.groupby('trainId').mean(...))
plt.plot(scan.positions, pumped_scan, label='pumped')
```