profile_2d

May 3, 2025

[1]:
```
%matplotlib inline
import sys
import os
import time
import numpy
import fabio
import pyFAI
from pyFAI.test.utilstest import UtilsTest
import pyFAI.method_registry
import pyFAI.integrator.azimuthal
print(f"Python version: {sys.version}")
print(f"PyFAI version: {pyFAI.version}")
start_time = time.perf_counter()
```

```
Python version: 3.12.3 | packaged by Anaconda, Inc. | (main, Apr 19 2024,
16:50:38) [GCC 11.2.0]
PyFAI version: 2025.4.0-dev0
```

[2]:
```
pyFAI.method_registry.IntegrationMethod.list_available()
```

[2]:
```
['IntegrationMethod(1d int, no split, histogram, python)',
 'IntegrationMethod(2d int, no split, histogram, python)',
 'IntegrationMethod(1d int, no split, histogram, cython)',
 'IntegrationMethod(2d int, no split, histogram, cython)',
 'IntegrationMethod(1d int, bbox split, histogram, cython)',
 'IntegrationMethod(2d int, bbox split, histogram, cython)',
 'IntegrationMethod(1d int, full split, histogram, cython)',
 'IntegrationMethod(2d int, full split, histogram, cython)',
 'IntegrationMethod(2d int, pseudo split, histogram, cython)',
 'IntegrationMethod(1d int, no split, CSR, cython)',
 'IntegrationMethod(2d int, no split, CSR, cython)',
 'IntegrationMethod(1d int, bbox split, CSR, cython)',
 'IntegrationMethod(2d int, bbox split, CSR, cython)',
 'IntegrationMethod(1d int, no split, CSR, python)',
 'IntegrationMethod(2d int, no split, CSR, python)',
 'IntegrationMethod(1d int, bbox split, CSR, python)',
 'IntegrationMethod(2d int, bbox split, CSR, python)',
 'IntegrationMethod(1d int, no split, CSC, cython)',
```

```
 'IntegrationMethod(2d int, no split, CSC, cython)',
 'IntegrationMethod(1d int, bbox split, CSC, cython)',
 'IntegrationMethod(2d int, bbox split, CSC, cython)',
 'IntegrationMethod(1d int, no split, CSC, python)',
 'IntegrationMethod(2d int, no split, CSC, python)',
 'IntegrationMethod(1d int, bbox split, CSC, python)',
 'IntegrationMethod(2d int, bbox split, CSC, python)',
 'IntegrationMethod(1d int, bbox split, LUT, cython)',
 'IntegrationMethod(2d int, bbox split, LUT, cython)',
 'IntegrationMethod(1d int, no split, LUT, cython)',
 'IntegrationMethod(2d int, no split, LUT, cython)',
 'IntegrationMethod(1d int, full split, LUT, cython)',
 'IntegrationMethod(2d int, full split, LUT, cython)',
 'IntegrationMethod(1d int, full split, CSR, cython)',
 'IntegrationMethod(2d int, full split, CSR, cython)',
 'IntegrationMethod(1d int, full split, CSR, python)',
 'IntegrationMethod(2d int, full split, CSR, python)',
 'IntegrationMethod(1d int, full split, CSC, cython)',
 'IntegrationMethod(2d int, full split, CSC, cython)',
 'IntegrationMethod(1d int, full split, CSC, python)',
 'IntegrationMethod(2d int, full split, CSC, python)',
 'IntegrationMethod(1d int, no split, histogram, OpenCL, Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, no split, histogram, OpenCL, Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, no split, histogram, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, no split, histogram, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, bbox split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, bbox split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, no split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, no split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, bbox split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, bbox split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, no split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, no split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, full split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
```

```
 'IntegrationMethod(2d int, full split, CSR, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, full split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, full split, CSR, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, bbox split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, bbox split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, no split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, no split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, bbox split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, bbox split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, no split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, no split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(1d int, full split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(2d int, full split, LUT, OpenCL, Intel(R) OpenCL / Intel(R)
Core(TM) i5-4308U CPU @ 2.80GHz)',
 'IntegrationMethod(1d int, full split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)',
 'IntegrationMethod(2d int, full split, LUT, OpenCL, Intel Gen OCL Driver /
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved)']
```

[3]:
```
print(len(pyFAI.method_registry.IntegrationMethod.list_available()))
```

```
67
```

[4]:
```
ai = pyFAI.load(UtilsTest.getimage("Pilatus1M.poni"))
img = fabio.open(UtilsTest.getimage("Pilatus1M.edf")).data
ai
```

[4]:
```
Detector Pilatus 1M      PixelSize= 172µm, 172µm         BottomRight (3)
Wavelength= 1.000000e-10 m
SampleDetDist= 1.583231e+00 m   PONI= 3.341702e-02, 4.122778e-02 m
rot1=0.006487  rot2=0.007558  rot3=0.000000 rad
DirectBeamDist= 1583.310 mm     Center: x=179.981, y=263.859 pix        Tilt=
0.571° tiltPlanRotation= 130.640°  = 1.000Å
```

[5]:
```
kw1 = {"data": img, "npt":1000}
kw2 = {"data": img, "npt_rad":1000}
```

```
res = {}
for k,v in pyFAI.method_registry.IntegrationMethod._registry.items():
    print(k)
    if k.dim == 1:
        res[k] = %timeit -o ai.integrate1d(method=v, **kw1)
    else:
        res[k] = %timeit -o ai.integrate2d(method=v, **kw2)
```

```
Method(dim=1, split='no', algo='histogram', impl='python', target=None)
73.3 ms ± 10.2 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='no', algo='histogram', impl='python', target=None)
160 ms ± 14.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='histogram', impl='cython', target=None)
18.6 ms ± 101  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='histogram', impl='cython', target=None)
38.4 ms ± 399  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=1, split='bbox', algo='histogram', impl='cython', target=None)
45.5 ms ± 281  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='histogram', impl='cython', target=None)
65.6 ms ± 500  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=1, split='full', algo='histogram', impl='cython', target=None)
231 ms ± 4.45 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='full', algo='histogram', impl='cython', target=None)
351 ms ± 89.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='pseudo', algo='histogram', impl='cython', target=None)
506 ms ± 10.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csr', impl='cython', target=None)
11.3 ms ± 112  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='csr', impl='cython', target=None)
17.7 ms ± 3.69 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='csr', impl='cython', target=None)
13.8 ms ± 215  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='bbox', algo='csr', impl='cython', target=None)
17.9 ms ± 1.31 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csr', impl='python', target=None)
19.9 ms ± 254  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='no', algo='csr', impl='python', target=None)
29.5 ms ± 2.36 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='csr', impl='python', target=None)
22 ms ± 639  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='csr', impl='python', target=None)
30.7 ms ± 844  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csc', impl='cython', target=None)
13.3 ms ± 164  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='no', algo='csc', impl='cython', target=None)
20.4 ms ± 683  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='csc', impl='cython', target=None)
17.1 ms ± 127  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
Method(dim=2, split='bbox', algo='csc', impl='cython', target=None)
28.5 ms ± 707  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csc', impl='python', target=None)
16.7 ms ± 29.3  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='csc', impl='python', target=None)
29.1 ms ± 5.81 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='csc', impl='python', target=None)
22 ms ± 269  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='csc', impl='python', target=None)
32.9 ms ± 1.42 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='lut', impl='cython', target=None)
15.2 ms ± 401  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='bbox', algo='lut', impl='cython', target=None)
27.8 ms ± 1.11 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='lut', impl='cython', target=None)
12.2 ms ± 219  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='lut', impl='cython', target=None)
19.4 ms ± 278  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='lut', impl='cython', target=None)
15 ms ± 292  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='full', algo='lut', impl='cython', target=None)
29.7 ms ± 3.33 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csr', impl='cython', target=None)
14.9 ms ± 2.64 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='full', algo='csr', impl='cython', target=None)
16.8 ms ± 211  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csr', impl='python', target=None)
22 ms ± 253  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='full', algo='csr', impl='python', target=None)
29.4 ms ± 141  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csc', impl='cython', target=None)
23.8 ms ± 265  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='full', algo='csc', impl='cython', target=None)
32.4 ms ± 223  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csc', impl='python', target=None)
21.9 ms ± 189  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='full', algo='csc', impl='python', target=None)

WARNING:pyFAI.opencl.azim_hist:Apparently 64-bit atomics are missing on device
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz, possibly falling back on 32-bit
atomics (loss of precision) but it can be present and not declared as Nvidia
does

32.2 ms ± 340  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='histogram', impl='opencl', target=(1, 0))

/home/kieffer/.venv/py312/lib/python3.12/site-packages/pyopencl/cache.py:496:
CompilerWarning: Non-empty compiler output encountered. Set the environment
variable PYOPENCL_COMPILER_OUTPUT=1 to see more.
```

```
  _create_built_program_from_source_cached(
/home/kieffer/.venv/py312/lib/python3.12/site-packages/pyopencl/cache.py:500:
CompilerWarning: Non-empty compiler output encountered. Set the environment
variable PYOPENCL_COMPILER_OUTPUT=1 to see more.
  prg.build(options_bytes, devices)
WARNING:pyFAI.opencl.azim_hist:Apparently 64-bit atomics are missing on device
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz, possibly falling back on 32-bit
atomics (loss of precision) but it can be present and not declared as Nvidia
does

35.6 ms ± 5.48 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='no', algo='histogram', impl='opencl', target=(1, 0))

WARNING:pyFAI.opencl.azim_hist:Apparently 64-bit atomics are missing on device
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved, possibly falling back on
32-bit atomics (loss of precision) but it can be present and not declared as
Nvidia does

37.1 ms ± 817  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=1, split='no', algo='histogram', impl='opencl', target=(0, 0))

Beignet: "unable to find good values for local_work_size[i], please provide\n" "
local_work_size[] explicitly, you can find good values with\n" " trial-and-error
method."
WARNING:pyFAI.opencl.azim_hist:Apparently 64-bit atomics are missing on device
Intel(R) HD Graphics Haswell Ultrabook GT3 reserved, possibly falling back on
32-bit atomics (loss of precision) but it can be present and not declared as
Nvidia does

75.8 ms ± 425  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='no', algo='histogram', impl='opencl', target=(0, 0))
60.7 ms ± 164  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=1, split='bbox', algo='csr', impl='opencl', target=(1, 0))
25.6 ms ± 1.43 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=2, split='bbox', algo='csr', impl='opencl', target=(1, 0))
329 ms ± 2.86 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csr', impl='opencl', target=(1, 0))
17.8 ms ± 404  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='csr', impl='opencl', target=(1, 0))
316 ms ± 1.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='csr', impl='opencl', target=(0, 0))
28.4 ms ± 93.5  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='csr', impl='opencl', target=(0, 0))
62.3 ms ± 449  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='csr', impl='opencl', target=(0, 0))
20.5 ms ± 225  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='no', algo='csr', impl='opencl', target=(0, 0))
56.7 ms ± 938  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csr', impl='opencl', target=(1, 0))
25.5 ms ± 841  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
Method(dim=2, split='full', algo='csr', impl='opencl', target=(1, 0))
327 ms ± 3.14 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='csr', impl='opencl', target=(0, 0))
28.4 ms ± 111  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='full', algo='csr', impl='opencl', target=(0, 0))
62.3 ms ± 247  s per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='lut', impl='opencl', target=(1, 0))
21.7 ms ± 744  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='lut', impl='opencl', target=(1, 0))
302 ms ± 6.94 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='lut', impl='opencl', target=(1, 0))
16.5 ms ± 116  s per loop (mean ± std. dev. of 7 runs, 100 loops each)
Method(dim=2, split='no', algo='lut', impl='opencl', target=(1, 0))
208 ms ± 3.39 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='bbox', algo='lut', impl='opencl', target=(0, 0))
28.6 ms ± 169  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='bbox', algo='lut', impl='opencl', target=(0, 0))
592 ms ± 38.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='no', algo='lut', impl='opencl', target=(0, 0))
21.5 ms ± 208  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='no', algo='lut', impl='opencl', target=(0, 0))
381 ms ± 58.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='lut', impl='opencl', target=(1, 0))
31.8 ms ± 8.3 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='full', algo='lut', impl='opencl', target=(1, 0))
The slowest run took 4.35 times longer than the fastest. This could mean that an
intermediate result is being cached.
612 ms ± 365 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Method(dim=1, split='full', algo='lut', impl='opencl', target=(0, 0))
28.5 ms ± 123  s per loop (mean ± std. dev. of 7 runs, 10 loops each)
Method(dim=2, split='full', algo='lut', impl='opencl', target=(0, 0))
1.41 s ± 358 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[6]:
```
res
```

[6]:
```
{Method(dim=1, split='no', algo='histogram', impl='python', target=None):
<TimeitResult : 73.3 ms ± 10.2 ms per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='no', algo='histogram', impl='python', target=None):
<TimeitResult : 160 ms ± 14.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='histogram', impl='cython', target=None):
<TimeitResult : 18.6 ms ± 101  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='no', algo='histogram', impl='cython', target=None):
<TimeitResult : 38.4 ms ± 399  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
```

```
 Method(dim=1, split='bbox', algo='histogram', impl='cython', target=None):
<TimeitResult : 45.5 ms ± 281  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='histogram', impl='cython', target=None):
<TimeitResult : 65.6 ms ± 500  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=1, split='full', algo='histogram', impl='cython', target=None):
<TimeitResult : 231 ms ± 4.45 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='full', algo='histogram', impl='cython', target=None):
<TimeitResult : 351 ms ± 89.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='pseudo', algo='histogram', impl='cython', target=None):
<TimeitResult : 506 ms ± 10.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csr', impl='cython', target=None):
<TimeitResult : 11.3 ms ± 112  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='no', algo='csr', impl='cython', target=None):
<TimeitResult : 17.7 ms ± 3.69 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='csr', impl='cython', target=None):
<TimeitResult : 13.8 ms ± 215  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='bbox', algo='csr', impl='cython', target=None):
<TimeitResult : 17.9 ms ± 1.31 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csr', impl='python', target=None):
<TimeitResult : 19.9 ms ± 254  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='no', algo='csr', impl='python', target=None):
<TimeitResult : 29.5 ms ± 2.36 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='csr', impl='python', target=None):
<TimeitResult : 22 ms ± 639  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='csr', impl='python', target=None):
<TimeitResult : 30.7 ms ± 844  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csc', impl='cython', target=None):
<TimeitResult : 13.3 ms ± 164  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='no', algo='csc', impl='cython', target=None):
<TimeitResult : 20.4 ms ± 683  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='csc', impl='cython', target=None):
<TimeitResult : 17.1 ms ± 127  s per loop (mean ± std. dev. of 7 runs, 100 loops
```

```
each)>,
 Method(dim=2, split='bbox', algo='csc', impl='cython', target=None):
<TimeitResult : 28.5 ms ± 707  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csc', impl='python', target=None):
<TimeitResult : 16.7 ms ± 29.3  s per loop (mean ± std. dev. of 7 runs, 100
loops each)>,
 Method(dim=2, split='no', algo='csc', impl='python', target=None):
<TimeitResult : 29.1 ms ± 5.81 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='csc', impl='python', target=None):
<TimeitResult : 22 ms ± 269  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='csc', impl='python', target=None):
<TimeitResult : 32.9 ms ± 1.42 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='lut', impl='cython', target=None):
<TimeitResult : 15.2 ms ± 401  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='bbox', algo='lut', impl='cython', target=None):
<TimeitResult : 27.8 ms ± 1.11 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='lut', impl='cython', target=None):
<TimeitResult : 12.2 ms ± 219  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='no', algo='lut', impl='cython', target=None):
<TimeitResult : 19.4 ms ± 278  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='lut', impl='cython', target=None):
<TimeitResult : 15 ms ± 292  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='full', algo='lut', impl='cython', target=None):
<TimeitResult : 29.7 ms ± 3.33 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='csr', impl='cython', target=None):
<TimeitResult : 14.9 ms ± 2.64 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='full', algo='csr', impl='cython', target=None):
<TimeitResult : 16.8 ms ± 211  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='csr', impl='python', target=None):
<TimeitResult : 22 ms ± 253  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='csr', impl='python', target=None):
<TimeitResult : 29.4 ms ± 141  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='csc', impl='cython', target=None):
```

```
<TimeitResult : 23.8 ms ± 265  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='full', algo='csc', impl='cython', target=None):
<TimeitResult : 32.4 ms ± 223  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='csc', impl='python', target=None):
<TimeitResult : 21.9 ms ± 189  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='csc', impl='python', target=None):
<TimeitResult : 32.2 ms ± 340  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='histogram', impl='opencl', target=(1, 0)):
<TimeitResult : 35.6 ms ± 5.48 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='no', algo='histogram', impl='opencl', target=(1, 0)):
<TimeitResult : 37.1 ms ± 817  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=1, split='no', algo='histogram', impl='opencl', target=(0, 0)):
<TimeitResult : 75.8 ms ± 425  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='no', algo='histogram', impl='opencl', target=(0, 0)):
<TimeitResult : 60.7 ms ± 164  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=1, split='bbox', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 25.6 ms ± 1.43 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=2, split='bbox', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 329 ms ± 2.86 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 17.8 ms ± 404  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='no', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 316 ms ± 1.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 28.4 ms ± 93.5  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 62.3 ms ± 449  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 20.5 ms ± 225  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='no', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 56.7 ms ± 938  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
```

```
 Method(dim=1, split='full', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 25.5 ms ± 841  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='csr', impl='opencl', target=(1, 0)):
<TimeitResult : 327 ms ± 3.14 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 28.4 ms ± 111  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='csr', impl='opencl', target=(0, 0)):
<TimeitResult : 62.3 ms ± 247  s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 21.7 ms ± 744  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 302 ms ± 6.94 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 16.5 ms ± 116  s per loop (mean ± std. dev. of 7 runs, 100 loops
each)>,
 Method(dim=2, split='no', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 208 ms ± 3.39 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='bbox', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 28.6 ms ± 169  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='bbox', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 592 ms ± 38.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='no', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 21.5 ms ± 208  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='no', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 381 ms ± 58.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 31.8 ms ± 8.3 ms per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='lut', impl='opencl', target=(1, 0)):
<TimeitResult : 612 ms ± 365 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>,
 Method(dim=1, split='full', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 28.5 ms ± 123  s per loop (mean ± std. dev. of 7 runs, 10 loops
each)>,
 Method(dim=2, split='full', algo='lut', impl='opencl', target=(0, 0)):
<TimeitResult : 1.41 s ± 358 ms per loop (mean ± std. dev. of 7 runs, 1 loop
```

each)>}

[7]:
```
print(f"{'split':5s} | {'algo':9s} | {'impl':6s}| {'1d (ms)':8s} | {'2d (ms)':
 ↪8s} | {'ratio':6s} | Device")
print("-"*80)
for k in res:
    if k.dim == 1:
        k1 = k
        k2 = k._replace(dim=2)
        if k2 in res:
            print(f"{k1.split:5s} | {k1.algo:9s} | {k1.impl:6s}| {res[k1].
 ↪best*1000:8.3f} | {res[k2].best*1000:8.3f} | {res[k2].best/res[k1].best:6.
 ↪1f} | ",
                  end="")
            if k.target:
                print(pyFAI.method_registry.IntegrationMethod._registry.get(k).
 ↪target_name)
            else:
                print()
```

```
split | algo      | impl   | 1d (ms)   | 2d (ms)   | ratio  | Device
-------------------------------------------------------------------------------
no    | histogram | python|   65.402 |  139.494 |    2.1 |
no    | histogram | cython|   18.472 |   37.632 |    2.0 |
bbox  | histogram | cython|   45.160 |   65.100 |    1.4 |
full  | histogram | cython|  225.316 |  307.111 |    1.4 |
no    | csr       | cython|   11.137 |   15.278 |    1.4 |
bbox  | csr       | cython|   13.503 |   16.722 |    1.2 |
no    | csr       | python|   19.724 |   28.323 |    1.4 |
bbox  | csr       | python|   21.389 |   30.099 |    1.4 |
no    | csc       | cython|   13.081 |   19.941 |    1.5 |
bbox  | csc       | cython|   16.884 |   28.018 |    1.7 |
no    | csc       | python|   16.661 |   22.301 |    1.3 |
bbox  | csc       | python|   21.699 |   32.050 |    1.5 |
bbox  | lut       | cython|   14.706 |   27.162 |    1.8 |
no    | lut       | cython|   11.931 |   18.879 |    1.6 |
full  | lut       | cython|   14.508 |   27.532 |    1.9 |
full  | csr       | cython|   13.314 |   16.518 |    1.2 |
full  | csr       | python|   21.800 |   29.288 |    1.3 |
full  | csc       | cython|   23.498 |   32.143 |    1.4 |
full  | csc       | python|   21.673 |   31.875 |    1.5 |
no    | histogram | opencl|   30.926 |   36.327 |    1.2 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
no    | histogram | opencl|   75.367 |   60.464 |    0.8 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
bbox  | csr       | opencl|   24.877 |  325.822 |   13.1 | Intel(R) OpenCL /
```

```
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
no    | csr       | opencl|   17.438 |  313.680 |   18.0 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
bbox  | csr       | opencl|   28.272 |   61.680 |    2.2 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
no    | csr       | opencl|   20.282 |   55.421 |    2.7 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
full  | csr       | opencl|   24.879 |  324.234 |   13.0 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
full  | csr       | opencl|   28.249 |   62.000 |    2.2 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
bbox  | lut       | opencl|   21.047 |  294.854 |   14.0 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
no    | lut       | opencl|   16.242 |  202.838 |   12.5 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
bbox  | lut       | opencl|   28.460 |  562.594 |   19.8 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
no    | lut       | opencl|   21.338 |  321.616 |   15.1 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
full  | lut       | opencl|   22.269 |  316.551 |   14.2 | Intel(R) OpenCL /
Intel(R) Core(TM) i5-4308U CPU @ 2.80GHz
full  | lut       | opencl|   28.328 |  873.578 |   30.8 | Intel Gen OCL Driver
/ Intel(R) HD Graphics Haswell Ultrabook GT3 reserved
```

[8]:
```
print(f"Total runtime: {time.perf_counter()-start_time:.3f}s")
```

```
Total runtime: 241.364s
```