Using sharpen

Auditing the FFT grid with sharpen on Aggregate and Portfolio

Stephen J. Mildenhall

2026-10-30

The bucket sizer selects the bucket size b (`bs`) and the number of buckets $n = 2^{\ell}$, with ℓ = `log2`, from the analytic moments, before any FFT runs (Using FFT windows). The method `sharpen()` audits that choice afterward. It updates the object on neighboring grids and scores each by how well the computed law reproduces six analytic moments: the severity and aggregate mean, coefficient of variation (CV), and skewness. Each relative error e_i is divided by its validation tolerance τ_i (ϵ for a mean, 10ϵ for a CV, 100ϵ for a skewness, with ϵ = `validation_eps`), so a term above 1 fails validation. A moment that is zero or infinite in theory is skipped, and the score is the power mean of the m remaining terms,

$$s = \Big(\frac{1}{m}\sum_{i=1}^{m}\big(|e_i|/\tau_i\big)^2\Big)^{1/2}.$$

The window width is $W = b\,2^{\ell}$. A larger bucket gives a wider window and coarser resolution, so at fixed ℓ the score has one trough in b, and the probe is a line search: double b until the score stops improving, then halve it, at ℓ and $\ell - 1$, and at $\ell + 1$ only when nothing smaller meets the target (default 0.5). Each pair (b, ℓ) tried is a **cell**. A cell is *defective* when its law loses mass off the top of the window, $1 - \sum_k p_k > 10^{-12}$, and is never chosen while a sound cell exists. The winner is the best sound score among the cells that do not grow ℓ. A grid that already meets the target and holds its mass is not probed at all.

The method `sharpen`[1] is available on `Aggregate` and `Portfolio`. The probe records its outcome in the object's own program, so `build(a.program)` rebuilds the sharpened object. A moved grid becomes a `hints` clause, and a confirmed grid, or one whose recommendation was not executed, a `note`.

```
agg Cat 2 claims 1000 xs 0 sev lognorm 10 cv 10 poisson hints{log2=16; bs=1/16}
agg Fine 10 claims sev gamma 50 cv 0.5 poisson note{sharpen: grid confirmed, no change}
```

```
from aggregate import build, exhibits, __version__
from style import wprint, GT, refuse, figures
figures()
print(__version__)
```

```
1.43.1
```

[1] From aggregate 1.0

A grid that meets the target

The aggregate `Fine`, ten claims from a gamma severity, is sized well by default. Its score, 0.000113, is far under the target of 0.5 and nothing is lost off the top of the window, so no probe runs and the program gains a confirmation note.

```
f = build('agg Fine 10 claims sev gamma 50 cv 0.5 poisson')
f.sharpen()
wprint(f.sharpen_description)
wprint(f.program)
```

```
Sharpen: not run, the grid already scores 0.000113 against a target of 0.5.
agg Fine 10 claims sev gamma 50 cv 0.5 poisson note{sharpen: grid confirmed, no
change}
```

The string `sharpen_explanation` gives the same outcome in prose, together with the definition of the score. Its units are the validation tolerances, so a score of 1 is the boundary between passing and failing validation.

```
wprint(f.sharpen_explanation)
```

```
The sharpen score measures how well the realized grid reproduces the analytic
moments: severity and aggregate mean, CV and skewness, each relative error
divided by its own validation tolerance and combined in a power-2 norm. The
units are tolerance, so a score of 1 sits exactly on the validation pass
boundary and smaller is better. The target here is 0.5. The current grid scores
0.000113, at or under the target, and holds all of its mass, so no probe was run
and nothing was changed. Pass good_enough=0 to force the probe regardless.
```

A grid moved to a finer bucket

The aggregate `Cat` has two claims from a lognormal severity with CV 10, limited to 1000. The severity is bounded, so the sizer takes the limit times a high quantile of the claim count, about 12,000, as the window, and pads it to 16,384 at $b = 1/4$. That bucket is too coarse for a severity with a median near 1, and the default grid fails validation.

```
c = build('agg Cat 2 claims 1000 xs 0 sev lognorm 10 cv 10 poisson')
wprint(c.bs_description)
print(c.validation_description, c.validation_score)
```

```
bounded_small grid: bs=1/4, log2=16, window [0, 16,384] of width 16,384
fails sev mean, agg mean 2.8534000548938114
```

The probe moves the bucket from 1/4 to 1/16 at the same ℓ, and the score from 2.85 to 0.206. The aggregate then passes validation.

```
c.sharpen()
wprint(c.sharpen_description)
print(c.validation_description)
```

```
Sharpen: 11 cells in 0.30s, best score 0.206 vs 2.85 at the center, target 0.5.
1 better-scoring cell rejected as defective (mass off the end of the grid).
Moved: bs 1/4 to 1/16, at log2 16.
not unreasonable
```

The table `sharpen_df` has one row per cell, indexed by the steps `d_bs` and `d_log2` from the starting grid, and its `extent` column is the window width W. At $b = 1/32$ the score is better still, 0.048, but the window of width 2,048 loses 9.4×10^{-8} off the top, so the cell is defective and passed over.

```
cols = ['bs', 'log2', 'extent', 'score', 'deficit', 'defective', 'selected']
GT(c.sharpen_df[cols], formatters={'bs': '.4g', 'extent': ',.0f',
                                   'score': '.3f', 'deficit': '.1e'})
```

d_bs	*d_log2*	bs	log2	extent	score	deficit	defective	selected
-4	0	0.01562	16	1,024	4.004	2.2e-04	True	False
-3	-1	0.03125	15	1,024	4.027	2.2e-04	True	False
	0	0.03125	16	2,048	0.048	9.4e-08	True	False
-2	-1	0.0625	15	2,048	0.208	9.4e-08	True	False
	0	0.0625	16	4,096	0.206	1.1e-14	False	True
-1	-1	0.125	15	4,096	0.813	1.1e-14	False	False
	0	0.125	16	8,192	0.813	1.1e-16	False	False
0	-1	0.25	15	8,192	2.853	-4.4e-16	False	False
	0	0.25	16	16,384	2.853	-4.4e-16	False	False
1	-1	0.5	15	16,384	8.968	-2.2e-16	False	False
	0	0.5	16	32,768	8.968	2.2e-16	False	False

The `sharpen` exhibit, in its insurer view, shows the same table as a grid: bucket size down the rows, ℓ across the columns, and the score beside the defective flag. A ** marks the starting grid on both axes and surrounds the flag of the selected cell. A missing score or a blank flag marks a cell the line search did not reach.

```
GT(exhibits.exhibit_frames(c, 'sharpen', 'insurer')[0][1])
```

	score		defective	
bs	15	**16	15	**16
1/64	—	4.0041		True
1/32	4.0272	0.0480	True	True
1/16	0.2078	0.2063	True	**False**
1/8	0.8127	0.8127	False	False
**1/4	2.8534	2.8534	False	False
1/2	8.9675	8.9675	False	False

The sharpened program

The string `sharpen_program` is the sharpened program, with its `hints` clause, laid out one clause to a line. Building `c.program` reproduces the sharpened grid.

```
wprint(c.sharpen_program)
d = build(c.program)
print(d.bs, d.log2, d.validation_score)
```

```
agg Cat
  2 claims
  1000 xs 0
  sev lognorm 10 cv 10
  poisson
  hints{log2=16; bs=1/16}
0.0625 16 0.20633928386138262
```

With `execute=False` the probe only diagnoses the grid. The original grid is restored, and the recommendation goes into a `note` rather than a `hints` clause.

```
e = build('agg Cat 2 claims 1000 xs 0 sev lognorm 10 cv 10 poisson')
e.sharpen(execute=False)
wprint(e.sharpen_description)
print(e.bs, e.log2)
wprint(e.program)
```

```
Sharpen: 11 cells in 0.29s, best score 0.206 vs 2.85 at the center, target 0.5.
1 better-scoring cell rejected as defective (mass off the end of the grid).
Recommends bs 1/4 to 1/16, at log2 16; not executed, grid restored.
0.25 16
agg Cat 2 claims 1000 xs 0 sev lognorm 10 cv 10 poisson note{sharpen: probe not
executed, recommends log2 16 and bs 1/16}
```

Sharpening a Portfolio

On a `Portfolio` the call is the same and the score is computed for the total. Here two limited lognormal units on a common grid move from $b = 1/2$ to $1/16$, and the `hints` clause attaches to the portfolio program.

```
p = build('port Cats agg A 2 claims 1000 xs 0 sev lognorm 10 cv 10 poisson '
          'agg B 3 claims 500 xs 0 sev lognorm 20 cv 5 poisson')
p.sharpen()
wprint(p.sharpen_description)
print(p.validation_description)
wprint(p.program)
```

```
Sharpen: 12 cells in 0.26s, best score 0.0544 vs 2.98 at the center, target 0.5.
1 better-scoring cell rejected as defective (mass off the end of the grid).
Moved: bs 1/2 to 1/16, at log2 16.
not unreasonable
port Cats hints{log2=16; bs=1/16} agg A 2 claims 1000 xs 0 sev lognorm 10 cv 10
poisson agg B 3 claims 500 xs 0 sev lognorm 20 cv 5 poisson
```

Refused calls

The line search steps in factors of two, so the argument `bs_limit`, the largest factor by which it may move the bucket, must be a power of two. Any other value raises a `ValueError`.

```
refuse(lambda: c.sharpen(bs_limit=10))
```

```
bs_limit must be a power of two >= 2, got 10
```