LISE++ *cute* v.18.0.26: Reactions in Wedges

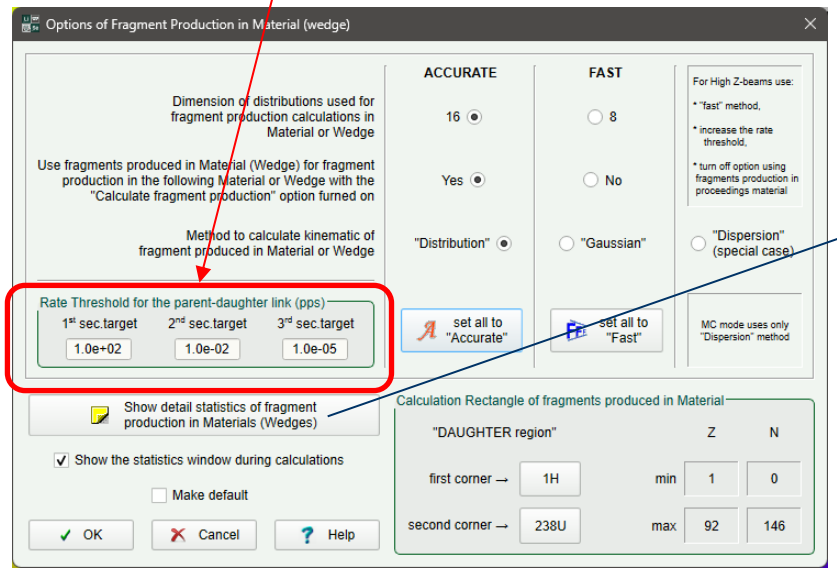
v.18.0.26 (v.18.1)
From 02/12/26

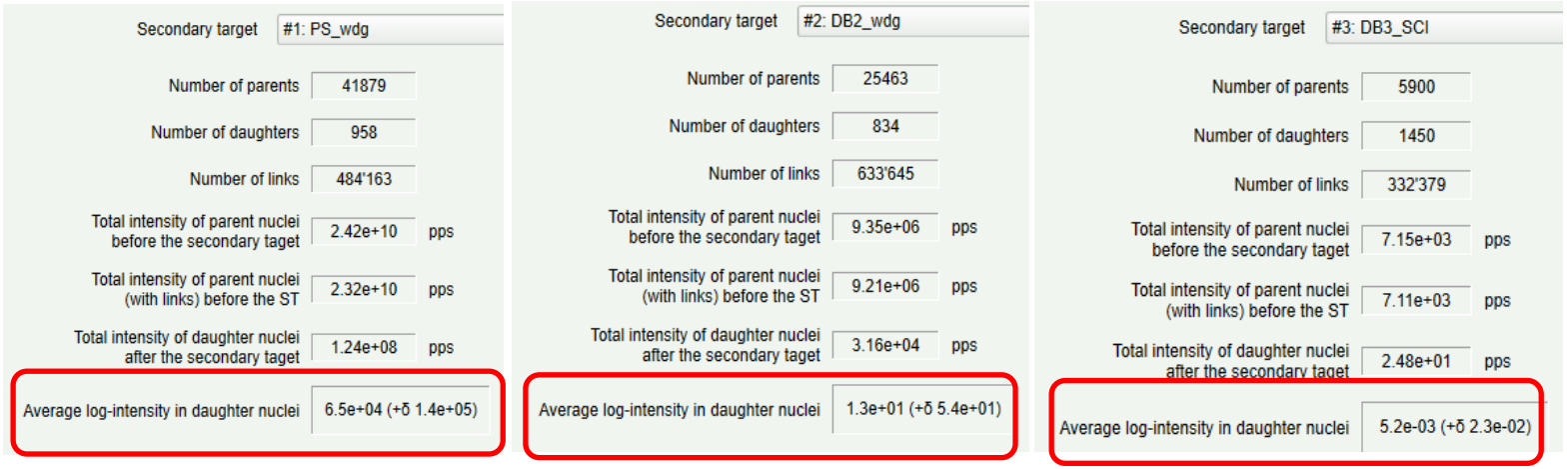
- **Bugs fixed:**
  - Reactions in wedge: **angular distributions** for fission products
  - Beam Dump : charge state plotting
  - Secondary target impact to power deposition analysis

- **Secondary Target** got a lot of attention
(speed, correctness, thresholds, statistics)
- **Power Deposition** utility was substantially expanded (per-reaction deposition, per-block reaction rates, corrected accounting, clearer formatting/notes)
- **UI/plotting polish** (BeamDump plotting, graph25 TOI color, discovery label rendering, d_CN1 dialog layouts)
- **Packaging/security** (satellite tools digitally certified)
- **Data update** (7 new isotopes around *Z*=60)

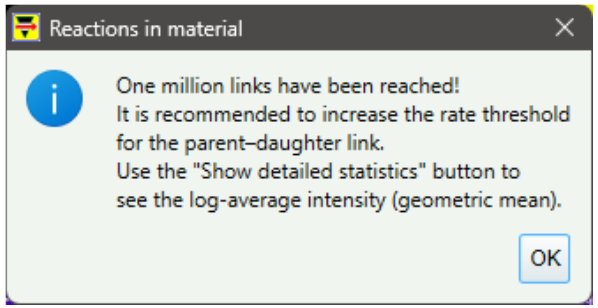
Fragment production in Materials (Secondary target)

New feature: Rate threshold per each target





New feature: average log-intensity info.
Recommended to use for threshold settings



- If you get the “million links” message, cancel the calculation.
- In **Fragment production in Material** dialog, click **Show detail statistics** to identify which secondary target exceeded the limit.
- Then use the **average log-intensity** information to increase that target’s threshold.

Power Deposition Utility Update

Besides POWER DEPOSITION [kW] and RATE ANALYSIS [pps]
the new table : **Rate analysis by reactions**

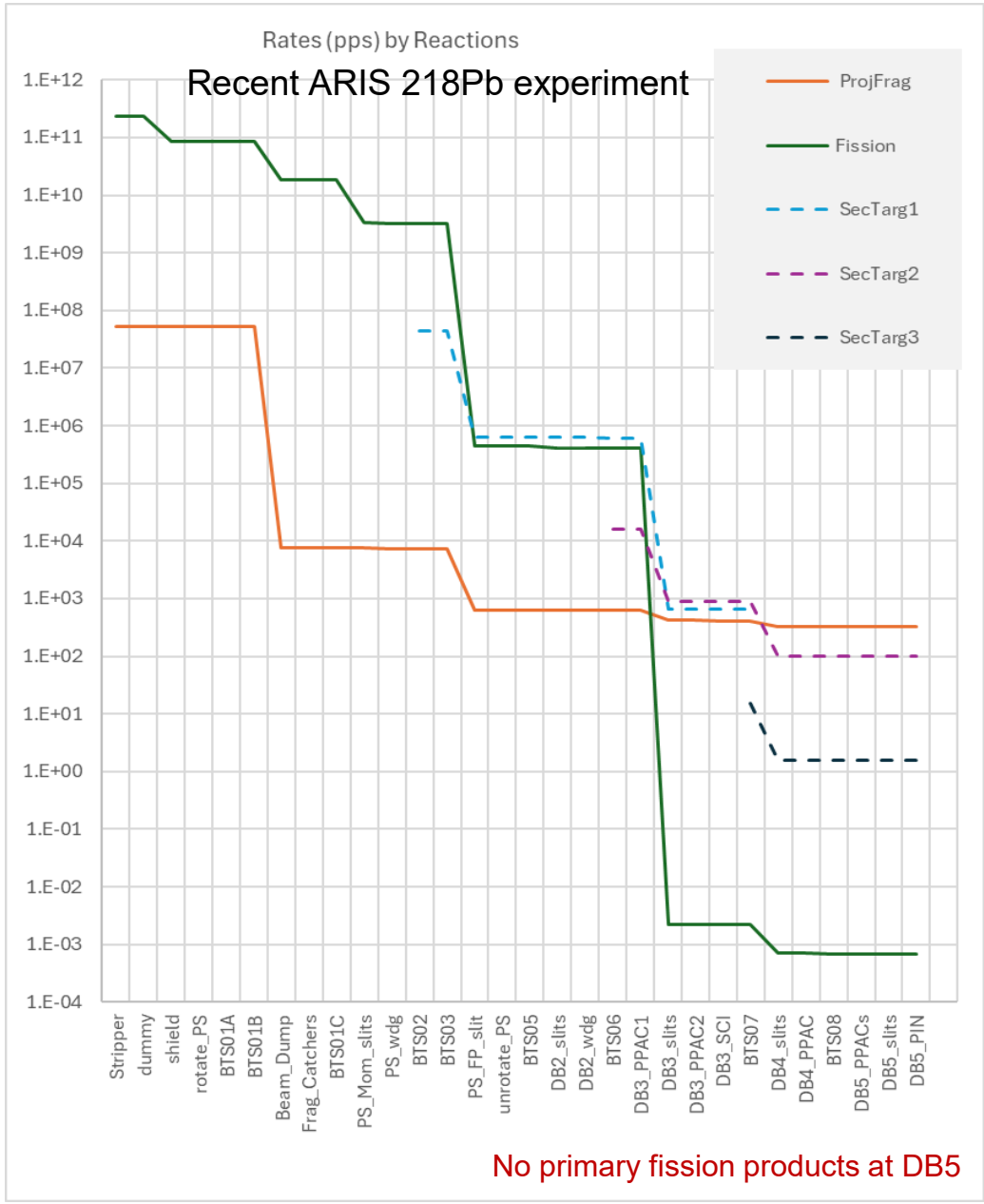
Power deposition in blocks and rate analysis: 220Pb_t2.0_PSw0.7_DB20.4_AA_Br4.84_EFF_FPinM_3.lpp

Save As | Print | PrintView | Search | Consolas | 9

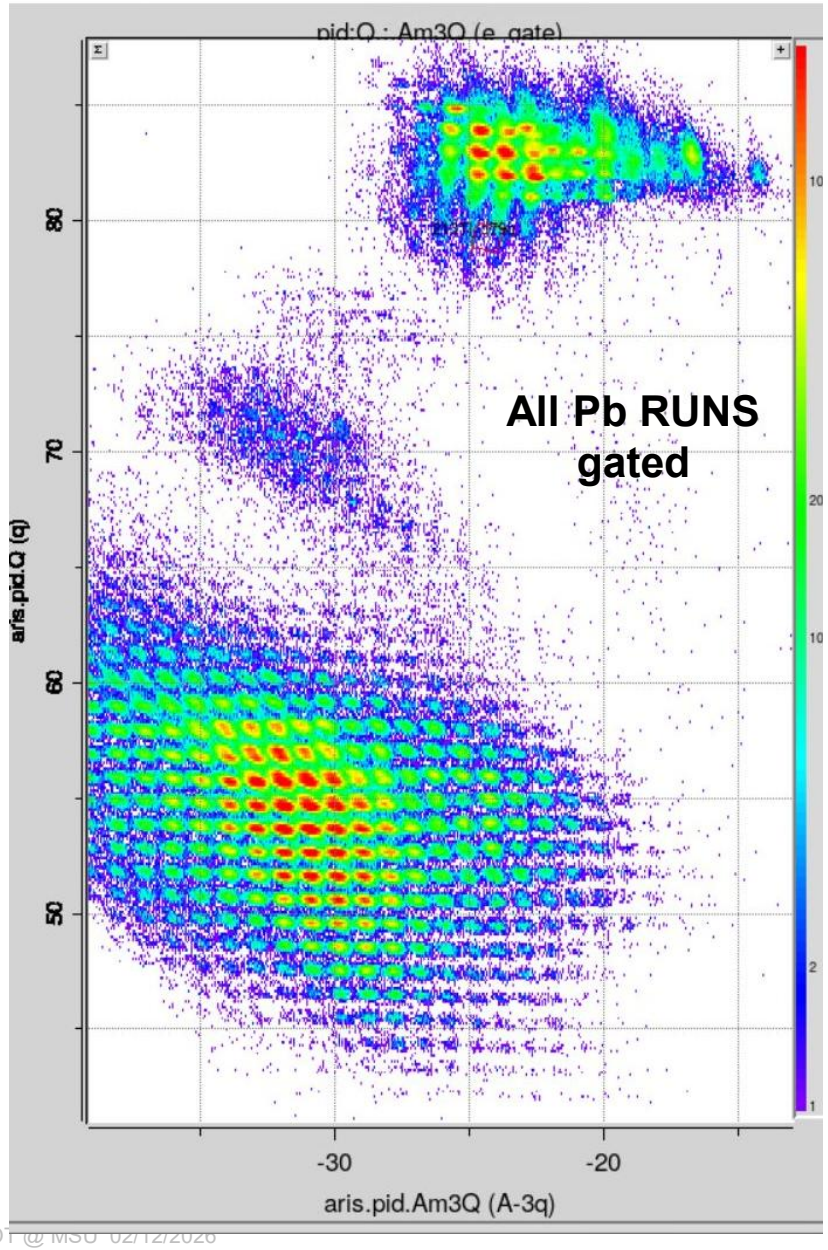
Rate analysis by reactions

#	Block	All frags	ProjFrag	AFlow	AFmid	AFhigh	SecTarg1	SecTarg2	SecTarg3
1	Target	2.31e+11	5.26e+07	4.25e+10	1.10e+11	7.92e+10	0	0	0
2	Stripper	2.31e+11	5.26e+07	4.25e+10	1.10e+11	7.92e+10	0	0	0
3	dummy	2.31e+11	5.26e+07	4.25e+10	1.10e+11	7.92e+10	0	0	0
4	shield	8.54e+10	5.26e+07	1.47e+10	3.86e+10	3.21e+10	0	0	0
5	rotate_PS	8.54e+10	5.26e+07	1.47e+10	3.86e+10	3.21e+10	0	0	0
6	BTS01A	8.54e+10	5.26e+07	1.47e+10	3.86e+10	3.21e+10	0	0	0
7	BTS01B	8.54e+10	5.26e+07	1.47e+10	3.86e+10	3.21e+10	0	0	0
8	Beam_Dump	1.86e+10	7.56e+03	6.15e+09	8.05e+09	4.44e+09	0	0	0
9	Frag_Catchers	1.86e+10	7.56e+03	6.15e+09	8.05e+09	4.44e+09	0	0	0
10	BTS01C	1.86e+10	7.56e+03	6.15e+09	8.05e+09	4.44e+09	0	0	0
11	PS_Mom_slits	3.30e+09	7.47e+03	8.59e+08	1.83e+09	6.17e+08	0	0	0
12	PS_wdg	3.26e+09	7.33e+03	8.47e+08	1.80e+09	6.09e+08	0	0	0
13	BTS02	3.20e+09	7.33e+03	8.18e+08	1.75e+09	5.91e+08	4.43e+07	0	0
14	BTS03	3.20e+09	7.33e+03	8.18e+08	1.75e+09	5.91e+08	4.43e+07	0	0
15	PS_FP_slit	1.07e+06	6.25e+02	4.04e+05	3.38e+04	2.44e+03	6.31e+05	0	0
16	unrotate_PS	1.07e+06	6.25e+02	4.04e+05	3.38e+04	2.44e+03	6.31e+05	0	0
17	correction_matrix	1.07e+06	6.25e+02	4.04e+05	3.38e+04	2.44e+03	6.31e+05	0	0
18	BTS05	1.07e+06	6.25e+02	4.04e+05	3.38e+04	2.44e+03	6.27e+05	0	0
19	DB2_slits	1.04e+06	6.24e+02	3.77e+05	3.23e+04	2.35e+03	6.27e+05	0	0
20	DB2_wdg	1.03e+06	6.18e+02	3.74e+05	3.20e+04	2.32e+03	6.22e+05	0	0
21	BTS06	1.03e+06	6.18e+02	3.66e+05	3.19e+04	2.32e+03	6.17e+05	1.63e+04	0
22	DB3_PPAC1	1.03e+06	6.18e+02	3.66e+05	3.19e+04	2.32e+03	6.17e+05	1.63e+04	0
23	DB3_slits	1.97e+03	4.18e+02	2.12e-03	1.27e-04	2.20e-06	6.50e+02	9.04e+02	0
24	DB3_PPAC2	1.97e+03	4.18e+02	2.12e-03	1.27e-04	2.20e-06	6.50e+02	9.04e+02	0
25	DB3_SCI	1.96e+03	4.15e+02	2.11e-03	1.26e-04	2.19e-06	6.47e+02	8.99e+02	0
26	BTS07	1.97e+03	4.15e+02	2.11e-03	1.26e-04	2.19e-06	6.45e+02	8.96e+02	1.51e+01
27	DB4_slits	4.30e+02	3.29e+02	6.61e-04	3.33e-05	6.56e-07	0	9.94e+01	1.57e+00
28	DB4_PPAC	4.30e+02	3.29e+02	6.61e-04	3.32e-05	6.56e-07	0	9.94e+01	1.57e+00
29	BTS08	4.30e+02	3.29e+02	6.56e-04	3.32e-05	6.56e-07	0	9.93e+01	1.57e+00
30	DB5_PPACs	4.30e+02	3.29e+02	6.56e-04	3.32e-05	6.56e-07	0	9.92e+01	1.57e+00
31	DB5_slits	4.30e+02	3.29e+02	6.56e-04	3.32e-05	6.56e-07	0	9.92e+01	1.57e+00
32	DB5_PIN	4.27e+02	3.27e+02	6.52e-04	3.30e-05	6.52e-07	0	9.87e+01	1.56e+00
33	DB5_TKE_SCI	1.20e-05	0	0	0	0	0	0	1.20e-05

Note: The leakage term in power likely arises from an inconsistency between
the primary-beam total reaction cross section and the summed isotope-production cross sections



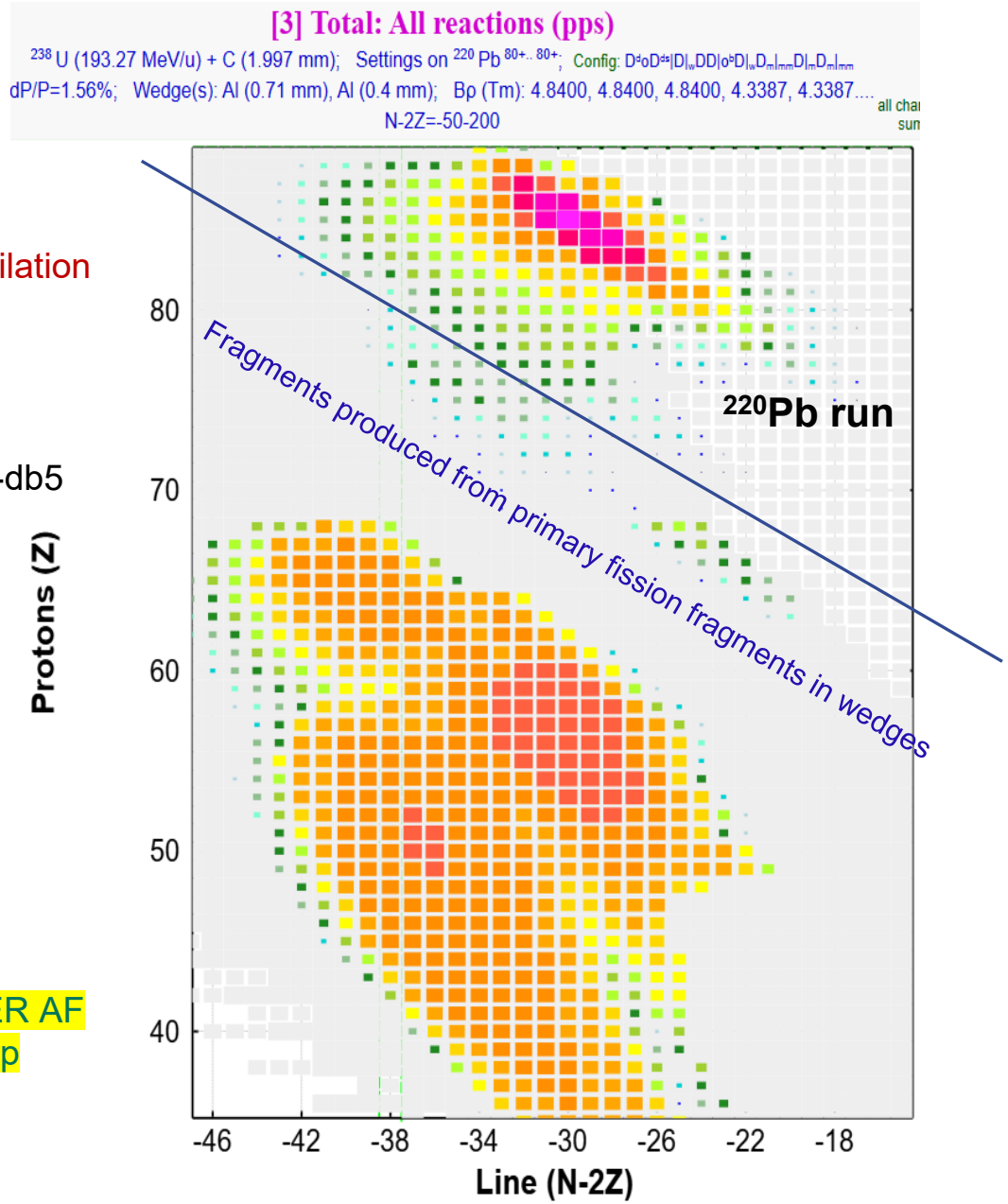
Pb runs: measured vs. calculated



The Experimental plot is compilation of different settings and was gated on

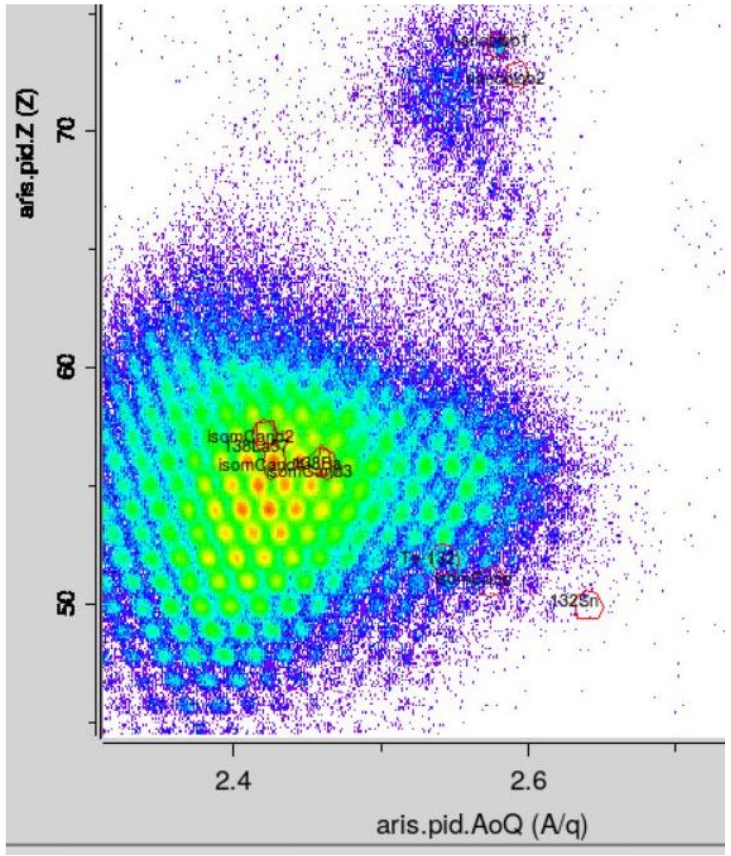
- He-like for high Z
- Fully-stripped for middle Z
- “no charge change” in db3-db5

Abrasion-Ablation and 3EER AF were applied for 1st step

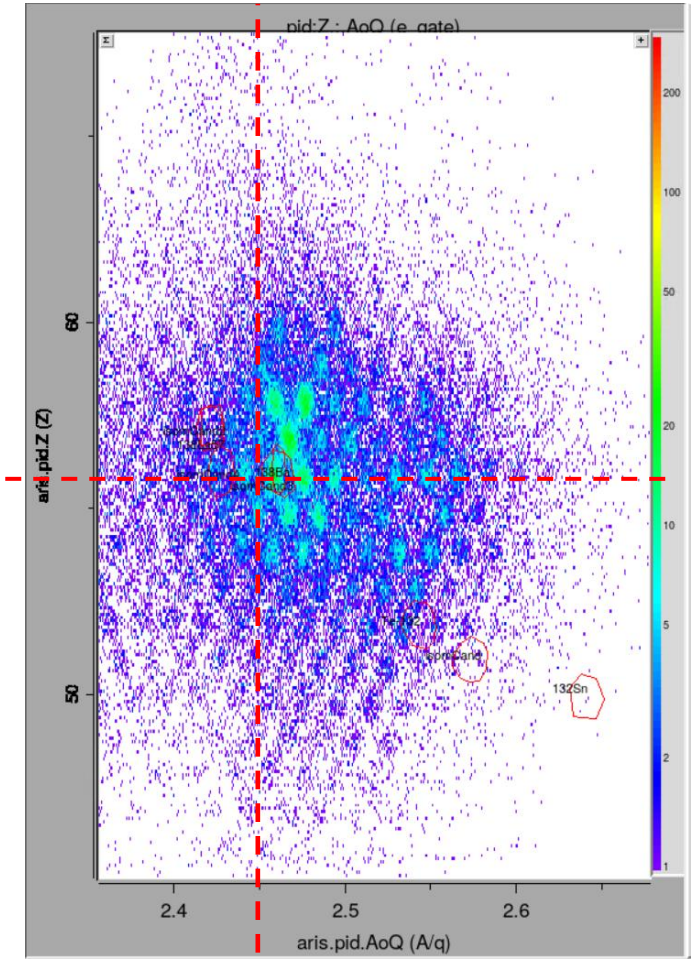


Pb runs: measured vs. calculated

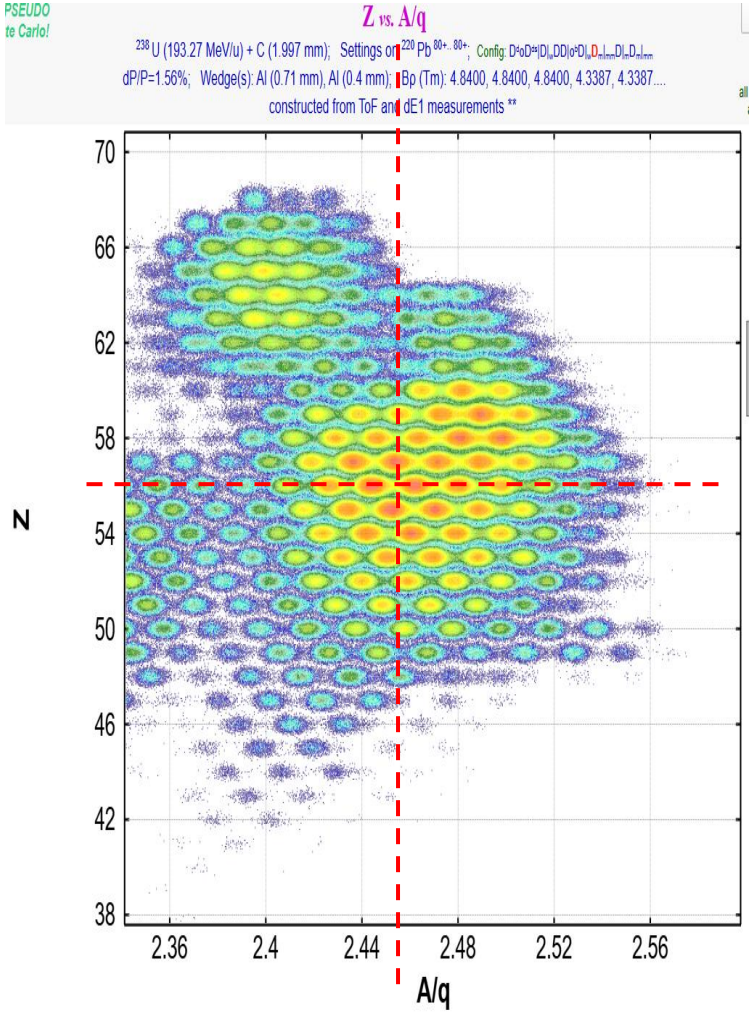
All RUNS
gated



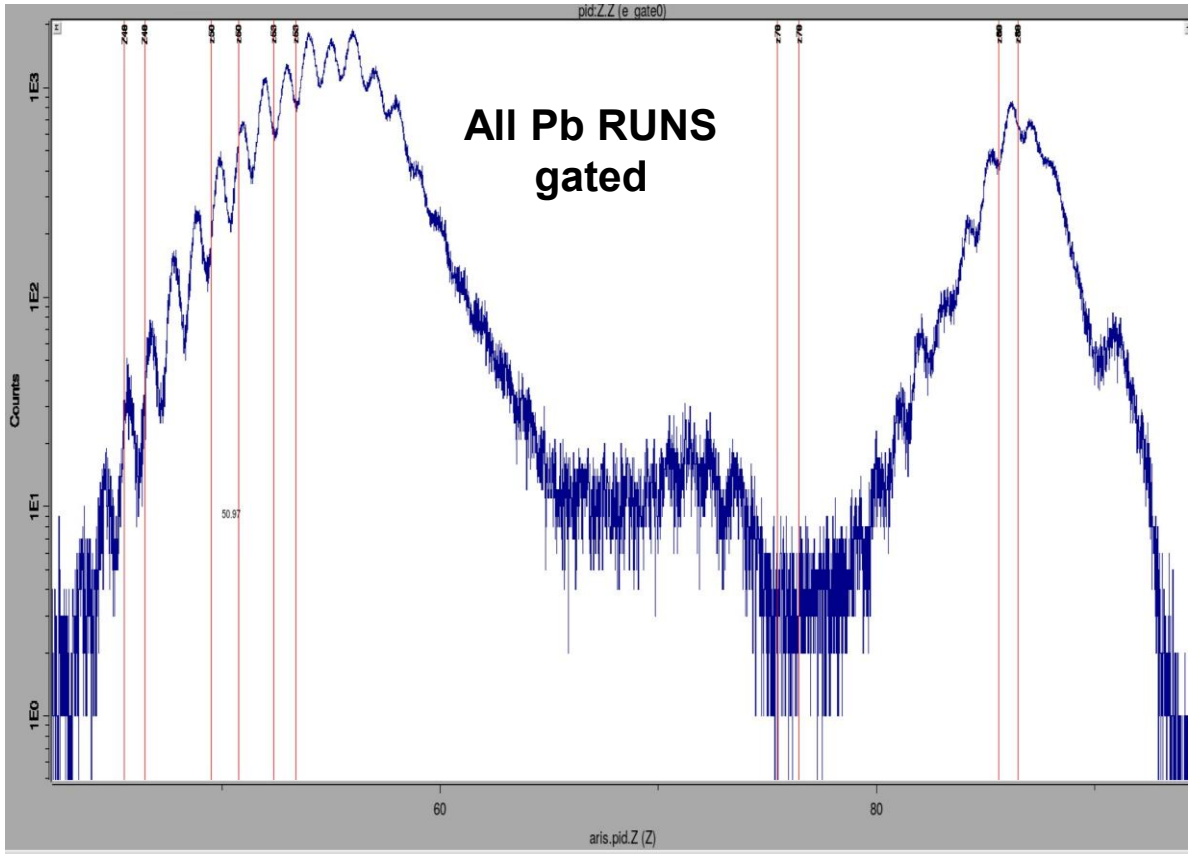
^{220}Pb RUNS
No gates

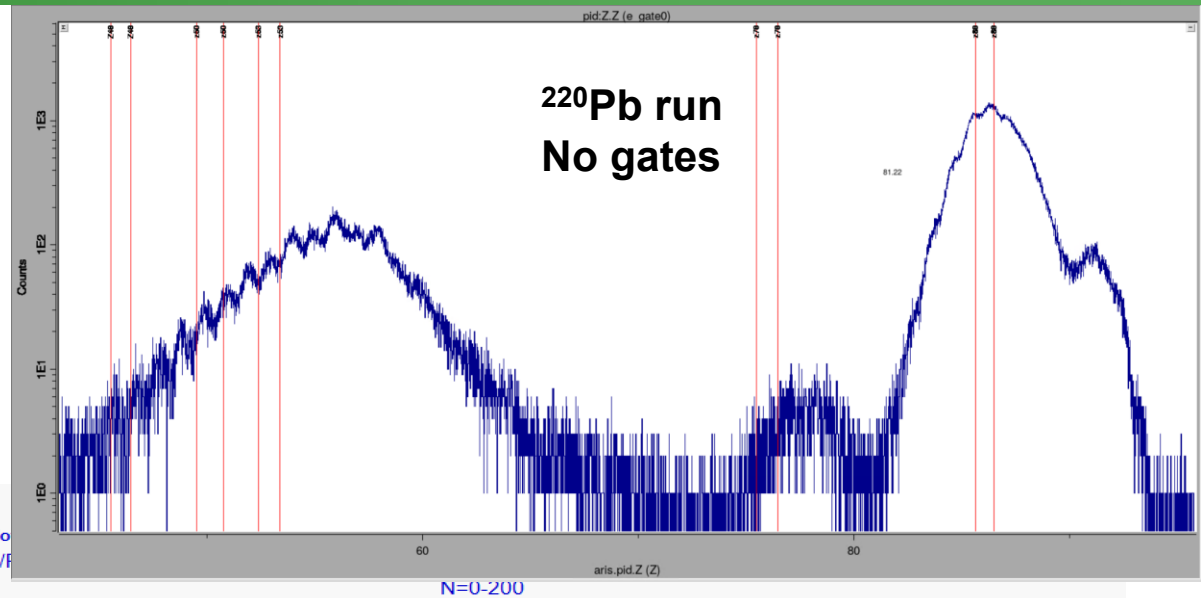


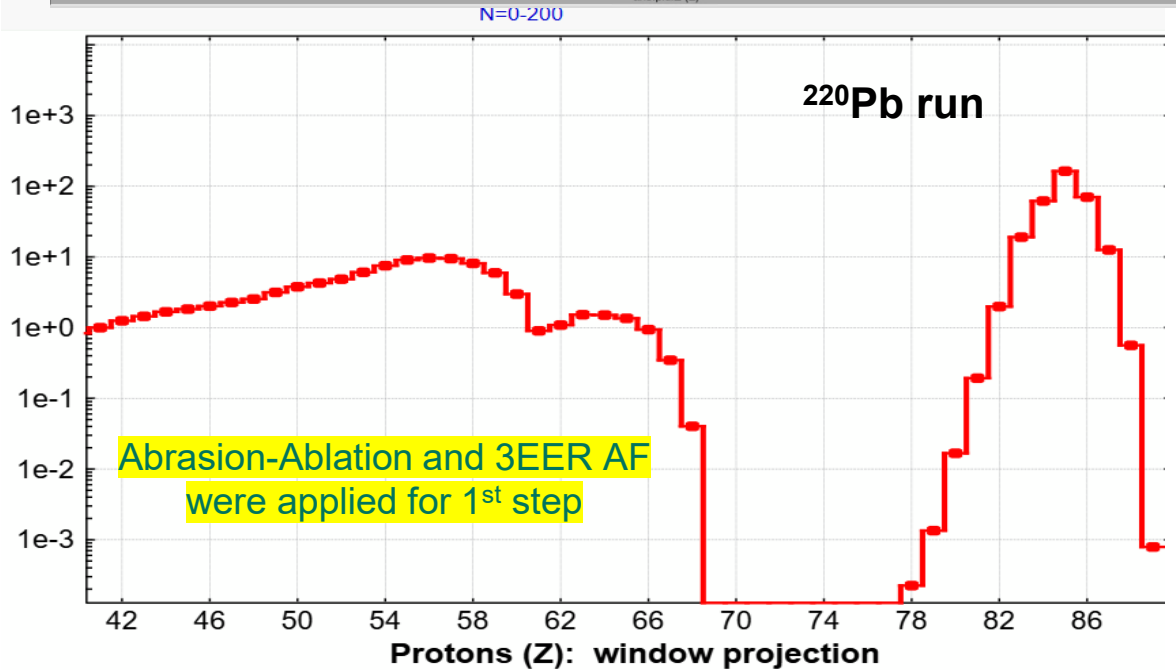
^{220}Pb RUNS
No gates



Pb runs: measured vs. calculated







The Experimental plot is compilation of different settings and was gated on

- He-like for high Z
- Fully-stripped for middle Z
- “no charge change” in db3-db5

Summary of 18.0.* versions

Plotting & labels
•**18.0.1** BeamDump 2D plotting fix for high power threshold
•**18.0.3** TOI color fix in graph25 (note about possibly changing default TOI background)
•**18.0.4** Discovery label rendering fix (skip unbound isotopes, smaller rectangle)

New data
•**18.0.2** Added 7 new isotopes (Sumikama, RIKEN, Z=60)

Packaging / signing
•**18.0.5** Satellite packages (PACE, GEMINI, Global, Charge, Kantele) digitally certified

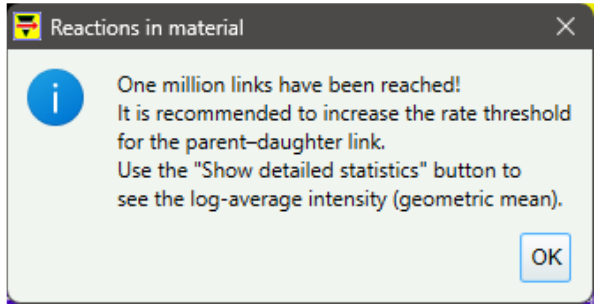
Secondary Target + angular module (performance + bug fixes)
•**18.0.6** Secondary target statistics update
•**18.0.7** Angular function optimized (≈2× faster)
•**18.0.8** Fixes in secondary target angular distribution module
•**18.0.9** Angular calculation bug fix (sigma_parallel)
•**18.0.10** Enforced minimum value for sigpar in `D2calcArray::Fill`
•**18.0.12** Personal threshold per each secondary target
•**18.0.21** Added `std::tuple<double,double,double> D2calcArray::calculateAverageRate() const`
•**18.0.22–18.0.24** Secondary target statistics/debug now use average log-intensity and include log/asym-error handling, plus updated “Million links” wording

Beam.PhysCalculator UI
•**18.0.11** Layout updates in d_CN1 dialogs

Power Deposition utility (major functional expansion)
•**18.0.13–18.0.15** Introduced `ResidDepositionJoint` per reaction per block, integrated it into calculations, plus memory/time optimizations and header cleanup
•**18.0.16–18.0.18** Added/finished per-block reaction rate output, updated value formatting
•**18.0.17** Corrected secondary target contribution in total deposition
•**18.0.19** Added note about possible inconsistency in the power deposition window
•**18.0.25** Deposition loop exit condition now based on **rates**, not **power**
•**18.0.26** Correction in deposition for secondary-target rates

API / function interface change
•**18.0.20** Updated argument order in asym error functions

Reactions in Wedges vs. RAM



I used 3 wedges there reactions took place.
1.134 e6 2.5e6 1.18e6 links per wedge during calculation LISE used ~ 10 GB of RAM.
After calculations stopped -- 659.7 MB why decreased?
At closing RAM ~ 3 GB

That drop is normal — you're seeing **peak working data** during the link-building, then most of it gets **freed/trimmed** when the calculation phase ends.
What likely happened:
•**During the run (~10 GB):** LISE had to hold big, temporary structures:
- the parent–daughter *link list/graph* (millions of entries, plus container overhead)
- sorting / grouping buffers (often a second copy of indices/keys)
- hash tables / maps while merging links
- per-wedge temporary arrays (and sometimes "all wedges at once" if results are accumulated before final reduction)

•**After it stops (~660 MB):** those large structures are **cleared/destroyed**, so:
- the allocator returns large blocks to the OS (especially big allocations)
- Windows may also **trim the working set** when the app goes idle
- some memory stays because it's your baseline (Qt, tables, UI, caches, loaded data)

Also, it depends *what* you read in Task Manager:
•**Working set** (RAM currently resident) can drop a lot
•**Commit/Private bytes** may stay higher (heap reserved but not resident)
•Freed memory may become **Standby cache**, so system "free" doesn't jump as much even though your process shrank
If you want to confirm, check **Process Explorer** (or VMMap) and compare:
•*Private Bytes* vs *Working Set* before/after
If you want LISE to actively return memory between wedges, the usual pattern is to force containers to release capacity (example ideas):
•`std::vector<T>().swap(v);` after you're done
•`QHash<K,V>().swap(h);` / `QVector<T>().squeeze();` (Qt)
But bottom line: **RAM decreased because the big link-building data structures were temporary and got freed** (plus OS trimming).

Got it — if there was **no crash**, then a RAM jump **during close** is still very plausible and (annoyingly) pretty common on Windows.

The two usual reasons are:

1) Windows "working set" was trimmed after the calculation

After the heavy run, Windows often pushes a lot of your process pages out of RAM (standby/pagefile) and your **Working Set** drops (your ~660 MB).

When you close the file, your code often **walks large containers to free them** (destructors / `clear()`), especially if you have node-based containers like `std::list`, `QList`, `QMap`, `QHash`. That traversal **touches memory pages**, so Windows pages them back in → **Working Set spikes** (your ~3 GB), and then the process exits and everything is released.

This is *very* consistent with "no crash, spike happens only while closing".

2) Close triggers a "finalize/save" step that allocates temporarily

E.g. building a big `QString/QByteArray/QJsonDocument` for export, generating statistics, sorting, or building a report *before* writing.