TECHNICAL UPDATE

GLOBAL Charge-State Optimization in LISE^{++} transmission calculations

Oleg B. Tarasov
tarasov@frib.msu.edu

Suppression pruning, exact transition reuse, and cache strategy

Release	Date	Component	Status
LISE++ 18.7.8	27 Aug 2026	Charge-state optimization	Implemented and release-built

Outcome Repeated non-equilibrium GLOBAL transitions are now calculated once per optical step and incoming charge state. Suppressed branches stop before further work, while equilibrium TabCharge behavior and the established 3% energy-match rule remain unchanged.

Measured performance For a 208Pb non-equilibrium local-region calculation with five charge-state locations, LISE++ 18.7.8 reduced runtime from 182.2 s to 93.3 s: 88.9 s saved, a 48.8% runtime reduction, or **1.95x faster** than version 18.7.7.

1. Purpose and Scope

This update addresses the runtime cost of recursive charge-state optimization, with emphasis on GLOBAL calculations in non-equilibrium mode. The implementation reduces duplicate solver work without introducing approximate GLOBAL cache keys or reusing results across separate optimization runs.

The work also corrects propagation of the incoming charge state in recursive GLOBAL calculations and aligns the optimizer's GLOBAL mode decision with the same energy-boundary conditions used by the central charges() function.

2. User-Facing Suppression Model

The charge-state suppression dialog defines Individual and Total cutoffs separately for fragments and the primary beam. Probabilities are expressed on the 0-1 scale.

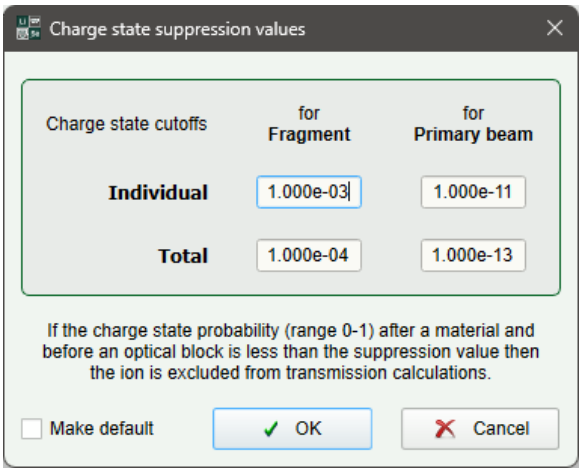

Figure 1. Charge-state suppression values used by the optimizer.

Cutoff	Applied to	Effect
Individual	Current charge-state probability q[i]	Rejects an individually negligible charge state.
Total	Cumulative path probability x q[i]	Rejects a path whose cumulative contribution is negligible.
Fragment / beam	Selected from scalc->Test(beam)	Uses the corresponding pair of dialog values.

A defensive total-cutoff guard now runs at recursive entry. When a path is already below its fragment or beam Total value, the function returns before charge-distribution or transmission work. Candidate children continue to require both Total and Individual tests before recursion.

3. Previous Behavior and Bottlenecks

- The recursive search recalculated non-equilibrium GLOBAL distributions for equivalent nodes reached through different earlier charge-state paths.

- The application-wide TabCharge cache was intentionally disabled for non-equilibrium GLOBAL calculations because the result depends on the incoming distribution and material thickness.
- The optimizer classified GLOBAL as non-equilibrium without checking emoy > UB_Global, even though charges() applies that upper-bound condition.
- At recursive steps, Qtab_global was seeded before q_last was assigned from the previously selected charge. The calculation could therefore begin from index zero instead of the actual incoming charge state.

4. Implemented Design

4.1 Two cache layers with different responsibilities

Layer	Valid domain	Key / lifetime	Decision
TabCharge	Equilibrium and other path-independent distributions	Existing method, projectile Z, energy window, target Z, method, solid flag; application lifetime	Retained
ChargeTransitionCache	Optimizer's definite-state, non-equilibrium GLOBAL transitions	Optical step + incoming electron-count index; one OptimumChargeState run	Added

TabCharge is still needed. It avoids repeated equilibrium distribution calculations throughout LISE++. It is not a safe generic cache for true non-equilibrium GLOBAL evolution because its historical key does not include the incoming distribution or complete material history.

The new local cache is safe in this optimizer because each recursive node explicitly seeds one incoming charge state. At a fixed optical step, the isotope, material sequence, and energy path are unchanged by earlier charge choices. The resulting transition is therefore deterministic for the incoming charge index.

4.2 Transition-cache layout

```
entry = step * Z + incoming_index
distribution[entry][0 ... Z-1]
valid[entry]
memory = NumQ * Z * Z * sizeof(double) + NumQ * Z * sizeof(bool)
```

The cache copies the exact calculated probability array. It performs no interpolation, energy rounding, or 3% matching. Allocation and destruction are scoped to OptimumChargeState(), preventing stale reuse after beamline, isotope, material, or option changes.

4.3 Optimized recursive flow

```
1. Select fragment or beam suppression limits.
2. Return if cumulative probability is below Total.
3. Determine GLOBAL non-equilibrium mode, including emoy > UB_Global.
4. Load (step, incoming charge) transition when available.
5. On a miss, evolve through charge-state materials and store the result.
6. Recurse only when both cumulative Total and Individual tests pass.
```

5. Avoided GLOBAL Calls

Below UB_Global, the optimizer no longer treats the complete material traversal as non-equilibrium. It calculates the final equilibrium distribution through the established charges(..., ApplyEquilibrium=true) path, enabling TabCharge and avoiding intermediate GLOBAL calls whose results would be overwritten by the next equilibrium material.

Above UB_Global, a cache miss still runs the complete existing non-equilibrium material sequence. A hit skips only duplicate charges() calls; energy-loss traversal and beamline block discovery still execute, preserving the outgoing energy and optical-step context.

6. Physics and Numerical Compatibility

Concern	Resolution
Cross-run staleness	Impossible: the transition cache exists only for one optimization call.
Approximate GLOBAL reuse	Not used: cache keys identify an exact step and incoming charge state.
Material evolution	A miss retains the original ordered sequence of charges() calls through all applicable materials.
Energy-boundary behavior	Non-equilibrium selection now matches charges(): GLOBAL method, non-ion-source reaction, EquilibriumMode == 1, Z >= MinZ_Global, and emoy > UB_Global.
TabCharge compatibility	Historical 3% matching and match-selection order are preserved.
Incoming charge	Qtab_global is now seeded with the actual previously selected charge index.

Intentional correction The q_last initialization fix can change results that previously started from the wrong incoming state. This is a physics correction, not a cache approximation.

7. Performance Characterization

7.1 Observed end-to-end benchmark

Case	18.7.7	18.7.8	Time saved	Improvement
208Pb, local region, non-equilibrium, 5 charge-state locations	182.2 s	93.3 s	88.9 s	48.8% less / 1.95x faster

This user-measured result demonstrates the improvement for the stated calculation and configuration. Runtime ratios can vary with hardware, charge-state survival, suppression thresholds, material layout, and the number of optical stages.

7.2 Evaluation-count model

The dominant improvement is a reduction in transition evaluations. If B charge states survive suppression at each of N optical stages, the prior recursive calculation evaluates one transition per non-terminal tree node. The new calculation evaluates at most one transition for each surviving incoming charge at each stage.

```
Previous evaluations:  1 + B + B^2 + ... + B^(N-1)
Cached evaluations:    1 + B * (N-1)
```

Representative case	Previous	Cached	Reduction	Avoided
4 stages, 4 surviving states	85	13	6.54x	84.7%

Each transition evaluation may contain multiple GLOBAL calls when several charge-state materials lie before the next optical block, so solver-call savings can exceed the transition-node count. Actual gains depend on suppression values, the number of optical stages, and how many charge states survive at each stage.

For the included high-Z smoke-test scale (Z = 75, NumQ = 4), the local cache occupies approximately 180,300 bytes, or 176.1 KiB.

8. Source Changes

File	Change
d_CN/d_Charge_optimum.cpp	Added ChargeTransitionCache, cutoff guard, shared suppression-limit selection, UB_Global-aligned mode selection, cache load/store, and corrected q_last initialization.
L_Calise/L_Charges_tab.cpp	Retained historical lookup semantics; corrected delete[] for both dynamically allocated arrays and removed obsolete commented alternative code.
L_Init/lise_version.h	Advanced LISE++ to Version 18.7.8 and added the release-note entry.
_install/LISE++.exe	Rebuilt MinGW release executable.

9. Build and Verification

Check	Result
Changed translation units	Compiled cleanly with the project's GNU++17 release flags.
Full release link	Succeeded; _install/LISE++.exe generated.
High-Z smoke test	AF_238U_Be_highZ.lpp completed in silent mode and wrote a populated four-stage .res5 transmission report.
Silent exit status	Exit code 2 is expected because MainWindow::closeEvent explicitly calls exit(2).
Patch hygiene	git diff --check completed without whitespace errors.

Build toolchain: Qt 6.7.0 with MinGW 11.2, release configuration Desktop_Qt_6_7_0_MinGW_64_bit-Release.

10. Maintenance Guidance

- Keep TabCharge enabled for equilibrium calculations; it remains valuable across repeated application calls.
- Do not extend TabCharge to non-equilibrium GLOBAL results unless the key includes the complete incoming distribution and all material-evolution inputs.
- Keep the optimizer-local cache scoped to one OptimumChargeState run unless explicit invalidation is added for isotope, beamline, material, and GLOBAL option changes.
- When modifying suppression logic, preserve both tests: Individual filters local probability, while Total filters cumulative path contribution.
- Benchmark future changes with representative high-Z, multi-stage cases because shallow or strongly suppressed trees naturally show fewer cache hits.