Efficient Storage and Compression of Intermediate Data in Lattice Calculations

Constantinos Costa (Rinnoco Ltd)*, Panos K. Chrysanthis(Rinnoco Ltd/University of Pittsburgh), Marios Costa(Cyprus University of Technology), Haralambos Panagopoulos(University of Cyprus)

RINNOCO
Research - Innovation - Communication

*costa.c@rinnoco.com
Co-founder & CEO,
Rinnoco Ltd
https://costadb.com

Cyprus University of Technology

University of Cyprus

July 27, 2026
LATTICE 2026

Co-funded by the European Union

Republic of Cyprus

RESEARCH & INNOVATION FOUNDATION

Outline

- Introduction & Motivation
- Efficient Workflows using Compression
  - Expression Shrinker Workflow (ESW)
  - Cache Compression Workflow (CCW)
- Command Line Interface
- Experimental Evaluation
- Conclusions & Future work

Problem

- Expressions **outgrow** ad hoc storage
  - **Perturbative calculations** with **stout-smearing** and **improved actions** generate intermediate expressions that are **large** and can be **reused**.

Millions of terms

Symbolic expansions can contain millions of algebraic terms before simplification.

Repeated reuse

The same intermediate output is loaded across several stages of the workflow.

Raw source pain

Plain Mathematica source is bulky, slow to parse, and memory-intensive to reload.

  - **The core problem** is not only file size. It is the **end-to-end workflow**: **store** the result, **load** it back into Mathematica, and optionally regenerate readable source.

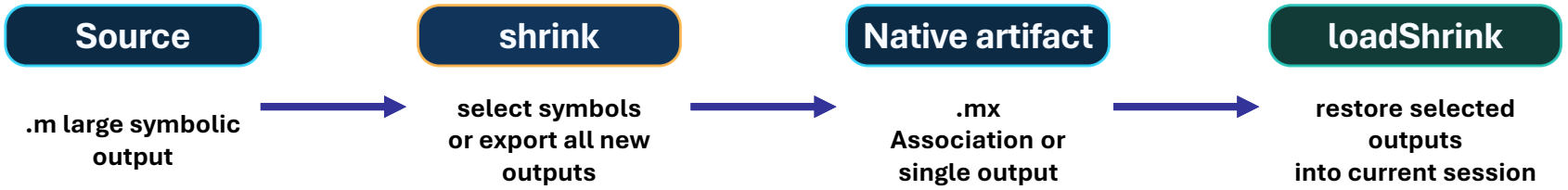
Goal: Make intermediate results **cheap** to **store, transfer, reload,** and **reproduce.**

RINNOCO

Expression Shrinker Workflow (ESW)

- Convert large source-generated expressions into ready to use Mathematica artifacts.

Key idea: artifact storage becomes a stable media **between expensive** symbolic **generation and easy reuse.**



Example 1: Shrink & Load Back

```
> shrink["SYM_GFSGg_lattice.m",
  "SYM_GFSGg_lattice.mx", intexpr]

> loadShrink["SYM_GFSGg_lattice.mx", intexpr]
```

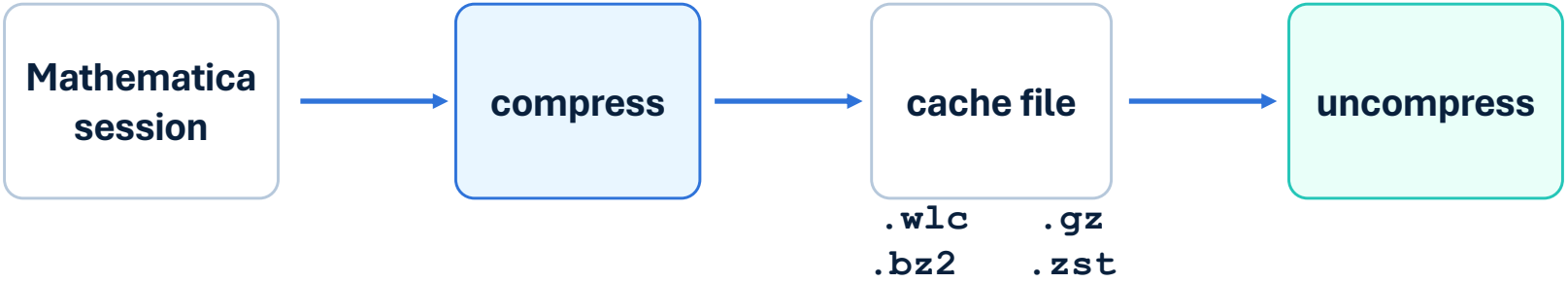
Example 2: Restore a shrunk file

```
> unShrink["SYM_GFSGg_lattice.mx",
 "SYM_GFSGg_lattice_restored.m",
  {intexpr, varlistGFSGg}]
```

RINNOCO

Cache Compression Workflow (CCW)

- Cache current in-memory symbols without returning to raw source



Example 3: Compress & Decompress the cache files

```
> compress[{intexpr, varlistGFSGg}, "SYM_GFSGg_lattice_cache"]

> uncompress["SYM_GFSGg_lattice_cache.wlc"]
```

  - The package detects new compressed file schemes automatically (using the extension) and stored payload format, while preserving support for older caches.

Command Line Interface (CLI)

- File-backed artifacts and in-memory caches share a small vocabulary

shrink	Source file → `.mx` artifact	Stores selected or newly defined outputs
loadShrink	`.mx` artifact → session	Restores all or selected targets
unShrink	`.mx` artifact → readable source	Writes wrapped multi-line InputForm
compress	session symbols → cache	WLC, GZIP, BZIP2, or ZSTD file
uncompress	cache → session	Auto-detects scheme when possible

The code is available on GitHub under the SUPER_QUANTA project:

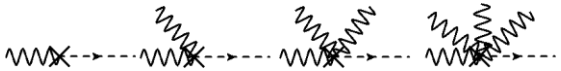
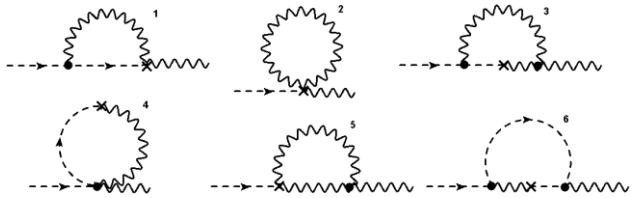
The project EXCELLENCE/0524/0208 is implemented under the programme of social cohesion “THALIA 2021-2027" co-funded by the European Union, through Research and Innovation Foundation.

https://github.com/Rinnoco/super_quanta/

RINNOCO

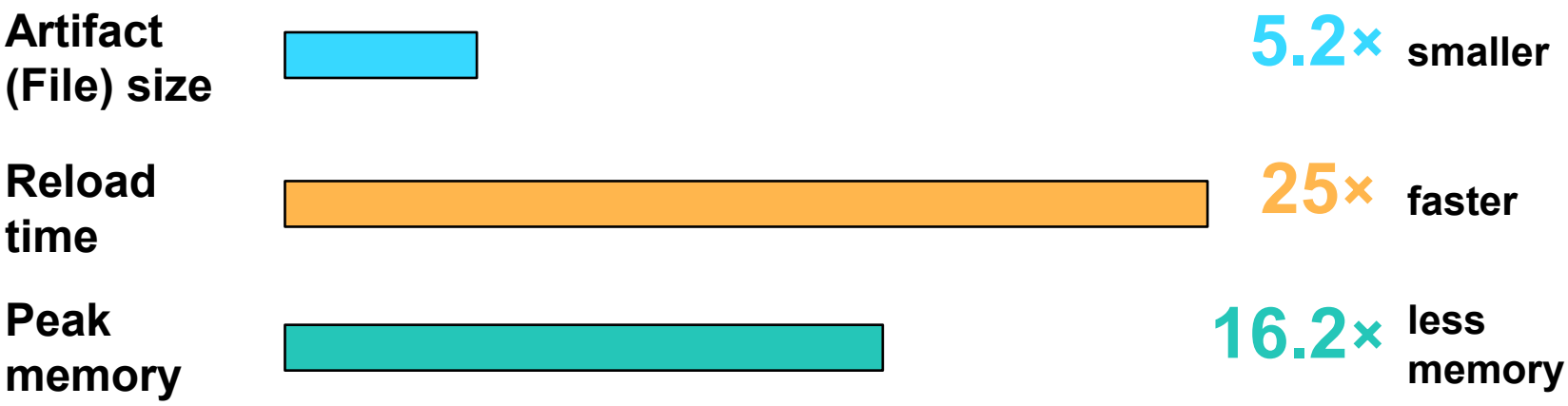
Experimental Setup

- Every operation executed in a fresh wolframscript kernel
- 5 measured runs per case / 1 warmup run per case
- Metrics: median wall time, artifact size, peak memory
- Schemes compared: .mx, raw serialized, Compress, GZIP, BZIP2, ZSTD

#	Description	Size	Lines
D1	**Supercurrent operator dataset:** The supercurrent operator is defined as $S_\mu = -\frac{1}{2}\,\mathrm{Tr}\left(\tilde{\bar{F}}_{\rho\sigma}\,[\gamma_\rho,\gamma_\sigma]\gamma_\mu\lambda\right)$, where the gauge links entering the definition of the operator are stout smeared. **Content:** Operator vertices  Four vertices for the operator S_μ. Wavy lines denote gluons (u_ν), while dashed lines denote gluinos (λ). The cross indicates the insertion of the supercurrent operator S_μ.	170.5 MB	2.81M
D2	**Green's Functions dataset:** **Content:** Main lattice convergent integrand arising from the one-loop diagrams contributing to the two-point Green's function of the supercurrent operator, together with the temporary-variable/tag list containing 809 unique structures.  One-loop Feynman diagrams contributing to the 2-pt Green's function $\langle u_\nu(q_1)\ S_\mu(x)\ \bar{\lambda}(q_2)\rangle$.	440.5 MB	6.77M

RINNOCO

Experimental evaluation



The gain is not just disk space: **faster**, **lower-memory** reloads make caching a practical design choice for **large perturbative calculations**.

- Shrinker Workflow:
  - **load_source**: load original **.m**
  - **shrink**: write **.mx**
  - **load_mx:** load shrunk **.mx**
  - **unshrink:** regenerate readable source

Experiments: Load & Memory Results

- Load Time

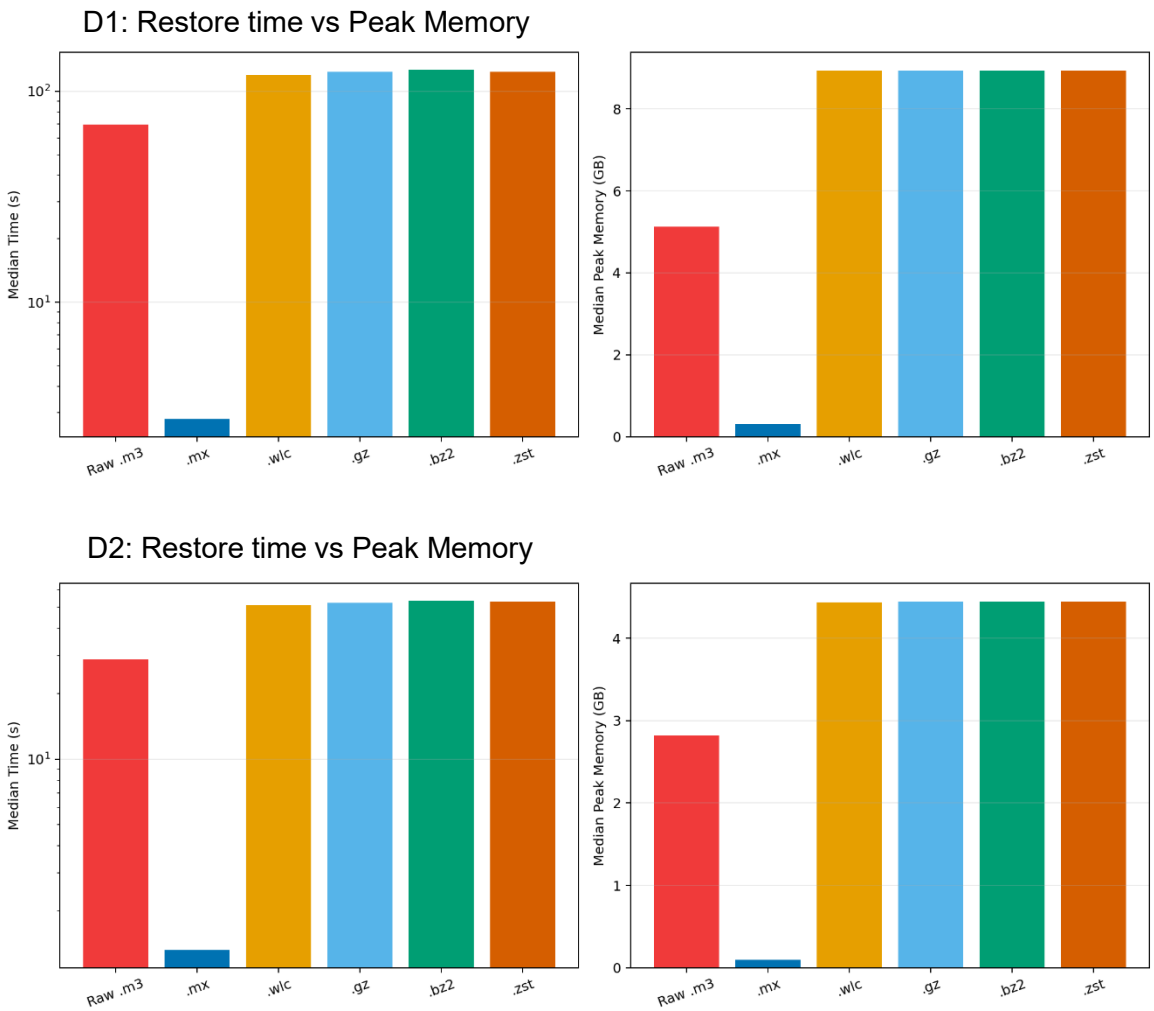
Dataset	Raw source	Shrinker (.mx) load	Speedup
D1	28.8 s	1.32 s	21.8x
D2	69.7 s	2.79 s	25.0x

- Memory

Dataset	Raw source	Shrinker Peak Memory (.mx)	Peak Memory Reduction
D1	2.82 GB	0.097 GB	29.1x
D2	5.13 GB	0.317 GB	16.2x

Experiments: Load Results (CCW)

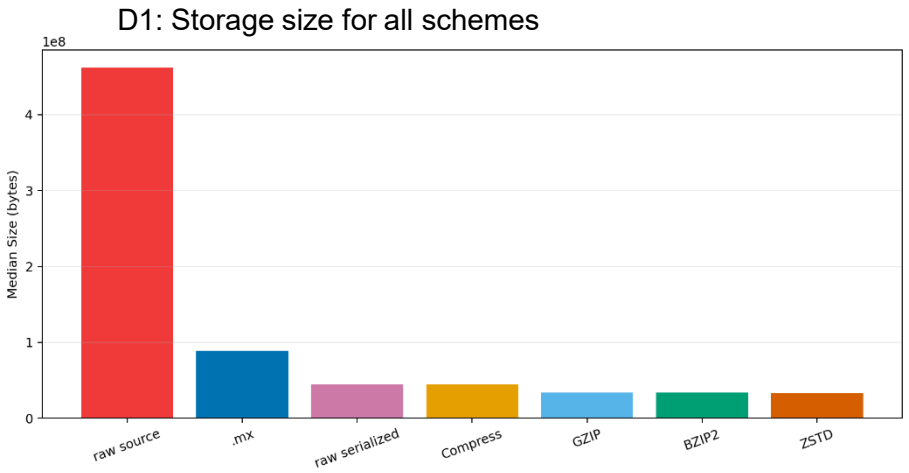
- **Shrinker (.mx)** restores data substantially **faster** while requiring **less memory** than **any** compressed-cache scheme, including the default raw-storage configuration.

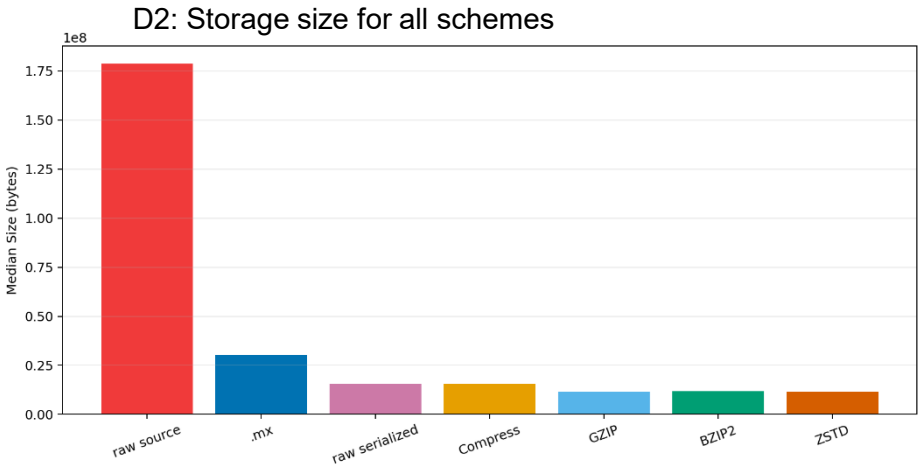


Experiments: Storage Results (CCW)

- **ZSTD** gives smallest files.
- **Shrinker (.mx)** gives fastest restoration.

Best choice depends on whether the bottleneck is **storage** or **reuse latency.**





Conclusions & Future Work

- Shrinker enables efficient storage and reuse of large Mathematica expressions.
  - Reduces file size, reload time, and peak memory use.
  - Supports native .mx files and multiple compression formats.
  - Improves reproducibility and workflow management in lattice perturbation theory.
- Extend support to numerical nonperturbative data.
  - Add automatic compression-method selection.
  - Support partial loading, remote storage, and HPC workflows.
  - Fix minor bugs for edge cases.

RINNOCO

Efficient Storage and Compression of Intermediate Data in Lattice Calculations

Constantinos Costa (Rinnoco Ltd)*, Panos K. Chrysanthis(Rinnoco Ltd/University of Pittsburgh), Marios Costa(Cyprus University of Technology), Haralambos Panagopoulos(University of Cyprus)

RINNOCO
Research - Innovation - Communication

*costa.c@rinnoco.com
Co-founder & CEO,
Rinnoco Ltd
https://costadb.com

Cyprus University of Technology

University of Cyprus

Thanks! Questions?

July 27, 2026
LATTICE 2026

Co-funded by the European Union

Republic of Cyprus

RESEARCH & INNOVATION FOUNDATION