Approachable Error
Bounded Lossy Compression

Robert Underwood, Ph.D. Candidate, School of Computing
About Me — Robert Underwood

• Ph.D. Candidate
• Computer Science
• Research Interests:
  • Lossy Compression
  • Reliability and Fault Tolerance
  • High Performance Computing

Personal Website
My CV
Exascale HPC Needs to Process Big Data

- Exascale Apps:
  - CESM-LE – 300TB/instance
  - HACC – 2000PB storage
- Exascale Instruments
  - LCLS-II – >250GB/s steaming

<table>
<thead>
<tr>
<th>System</th>
<th>Disk Storage</th>
<th>Peak Disk Bandwidth</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bebop</td>
<td>&lt; 2PB</td>
<td>n/a</td>
<td>0.3 PB</td>
</tr>
<tr>
<td>Mira</td>
<td>24 PB</td>
<td>n/a</td>
<td>~ 1 PB</td>
</tr>
<tr>
<td>Theta</td>
<td>11PB</td>
<td>n/a</td>
<td>~ 1 PB</td>
</tr>
<tr>
<td>Summit</td>
<td>250PB</td>
<td>2.5 TB/s</td>
<td>10PB</td>
</tr>
<tr>
<td>Aurora</td>
<td>230PB</td>
<td>25 TB/s</td>
<td>10PB</td>
</tr>
<tr>
<td>Projected Exascale</td>
<td>500 PB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Machine Characteristics from respective websites accessed 17 September 2020

Franck Cappello et al. “Use Cases of Lossy Compression for Floating Point Data in Scientific Datasets”. 2018
Compression is the Solution

Compression represents data in a more compact fashion

<table>
<thead>
<tr>
<th></th>
<th>Lossless</th>
<th>Lossy</th>
<th>Error Bounded Lossy Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>ZIP</td>
<td>JPEG</td>
<td>SZ/ZFP/MGARD</td>
</tr>
<tr>
<td>Compression Ratio</td>
<td>😒</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>😊</td>
<td>😒</td>
<td>??</td>
</tr>
<tr>
<td>Data Integrity</td>
<td>😊</td>
<td>😒</td>
<td>😊</td>
</tr>
</tbody>
</table>
Lossy Compression Is Not Approachable

- Too many interfaces
- Too difficult to configure
- Few tools to understand
- My dissertation provides a single interface to use, configure, understand compression
Outline

1. Introduction to Compression Principles
2. LibPressio
3. Automated Configuration of Compressors
4. Understanding the Effects of Compression
5. Conclusions and Future Work
Introduction to Error-Bounded Compression Principles

Lake Hartwell – Lossless (left), Lossy (right). The image on the right is 17 times smaller
SZ

- Prediction Based Compressor
  1. Data Prediction
  2. Linear Quantization
  3. Entropy Encoding
  4. Lossless Encoding

Di, Sheng and Cappello, Franck “Fast Error Bounded Lossy Compression with SZ” 2016
ZFP

• Transform Based Compressor
  1. Partition into grids of $4^n$
  2. Convert to fixed point by block
  3. Near Orthogonal Transform
     • Similar to JPEG Compression
  4. Bit manipulation

Lindstrom, Peter. “Fixed Rate Compressed Floating Point Arrays” 2012
MGARD

• Multi-Grid Method
  1. Determine Multi-grid coefficients
  2. Quantize “binding” coefficients
  3. Losslessly encode quantized coefficients

Whitney, Ben E. “Multilevel Techniques for Compression and Reduction of Scientific Data” 2018
A Generic Abstraction for the Compression of Dense Tensors
LibPressio Provides a Common Interface

- Common Abstractions for:
  - Loading compressors
  - Configuration
  - Compression/Decompression
  - Representing Data
  - Error Reporting
  - Computing Metrics

```c
//get the compressor
struct pressio_library = pressio_instance();

struct pressio_compressor* sz = pressio_get_compressor(library, "sz");

//configure, validate, and assign the options
struct pressio_options* config =
  pressio_compressor_get_options(sz);
pressio_options_set_integer(config, "sz:error_bound_mode", REL);
pressio_options_set_double(config, "sz:rel_err_bound", 0.01);
pressio_compressor_set_options(sz, config);

//read in an input buffer
size_t dims[] = {500,500,100};
struct pressio_data* description =
  pressio_data_new_empty(pressio_float_dtype, 3, dims);
struct pressio_data* input_data =
  pressio_io_data_path_read(description, "CLOUD48.bin.f32");

//create output buffers
struct pressio_data* compressed_data =
  pressio_data_new_empty(pressio_byte_dtype, 0, NULL);
struct pressio_data* decompressed_data =
  pressio_data_new_owing(pressio_float_dtype, 3, dims);

//compress and decompress the data
pressio_compressor_compress(sz, input_data, compressed_data);
pressio_compressor_decompress(sz, compressed_data,
  decompressed_data);
```

Get LibPressio
# Current Plugins and Tools

<table>
<thead>
<tr>
<th>Compressors</th>
<th>Meta-Compressors</th>
<th>Metrics</th>
<th>IO</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>• blosc</td>
<td>• linear_quantizer</td>
<td>• composite (lua)</td>
<td>• posix</td>
<td>• LibPressio-Tools</td>
</tr>
<tr>
<td>• imagemagick</td>
<td>• many_dependent</td>
<td>• error_stat</td>
<td>• csv</td>
<td>• LibPressio-Predict</td>
</tr>
<tr>
<td>• fpzip</td>
<td>• many_independent</td>
<td>• time</td>
<td>• hdf5</td>
<td>• LibPressio.jl</td>
</tr>
<tr>
<td>• mgard</td>
<td>• resize</td>
<td>• size</td>
<td>• select</td>
<td>• Z-Checker</td>
</tr>
<tr>
<td>• sz</td>
<td>• transpose</td>
<td>• no-op</td>
<td>• empty</td>
<td></td>
</tr>
<tr>
<td>• zfp</td>
<td>• opt</td>
<td>• kl_divergence</td>
<td>• no-op</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• no-op</td>
<td>• ks_test</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• fault_injector</td>
<td>• pearson</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• random_errors</td>
<td>• spatial_error</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ftk_critical_points</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• external</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bolded** plugins developed in collaboration with others
Meta Compressors Boost Productivity

• Not a compressor themselves

• Provide common services:
  • Auto Configuration tools
  • Pre/Post Processors
  • Parallel Runtimes
LibPressio Solves Problems

- Writing for multiple compressors is hard: over 100 bugs fixed to date
- Case Study: Z-Checker
  - Save over 1000 LoC (≈21%)
  - Better:
    - Over 10 new compressors
    - Over 3 new data formats
  - Faster: with MPI parallelism
  - Future proof: New compressors just need a recompile

Z-Checker Improvements

- Data Formats Supported
- Compressors Supported
- 1000s LoC
Future Work on LibPressio

- Support for accelerator sharing
  - GPUs
  - Threads
  - FPGAs
- Support for asynchrony/streams
- Support for sparse problems
Automated Configuration of Compressors
Automated Configuration Timeline

IPDPS 2020

FRaZ
- Bounding Compression Ratio

LibPressio-Opt
- Bounding User Metrics
- Performance Improvements

More Complex Apps
- Bounding Complex, Multi-faceted Metrics
1 - FRaZ: Fixed Ratio Compression

Can we tune compression using a control loop to bound the compression ratio?
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\tilde{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\tilde{c}; \theta_c))
\]
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\hat{c} \in U} Q \left( d_{f,t}, \tilde{d}_{f,t}(\hat{c}; \theta_c) \right)
\]
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\tilde{c} \in U} Q(d_f, t, \tilde{d_f}, t(\tilde{c}; \tilde{\theta_c}))
\]

Compression Ratio

Error Bound
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[ \max_{\hat{c} \in U} Q(d_f, t, \tilde{d}_f, t(\hat{c}; \theta_c)) \]
FRaZ: Approach

Formulate compressor configuration as an optimization problem

$$\max_{\vec{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\vec{c}; \vec{\theta}_c))$$

Compression Ratio

Data for a Field and Timestep

Error Bound

Allowed Error Bounds
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\vec{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\vec{c}; \theta_c))
\]

Compression Ratio
Data for a Field and Timestep
Error Bound
Allowed Error Bounds
Compression Function
FRaZ: Approach

Formulate compressor configuration as an optimization problem

\[ \max_{\mathbf{c} \in \mathcal{U}} Q(d_{f,t}, \widetilde{d}_{f,t}(\mathbf{c}; \mathbf{\theta}_c)) \]

- Compression Ratio
- Error Bound
- Data for a Field and Timestep
- Error Bound Mode
- Compression Function
- Allowed Error Bounds
FRaZ: Approach

- Why not use binary search?
  - It doesn’t work
  - The relationship between error bounds and compression ratios is not monotonic

- What can we use?
  - Optimization
    - Derivative Based Methods
      - Analytic Derivatives
      - Numerical Derivatives
    - Derivative Free Optimization
FRaZ: Approach

• Why not use binary search?
  • It doesn’t work
  • The relationship between error bounds and compression ratios is not monotonic

• What can we use?
  • Optimization
    • Derivative Based Methods
      • Analytic Derivatives (too challenging)
      • Numerical Derivatives
    • Derivative Free Optimization
FRaZ: Approach

• Why not use binary search?
  • It doesn’t work
  • The relationship between error bounds and compression ratios is not monotonic

• What can we use?
  • Optimization
    • Derivative Based Methods
      • Analytic Derivatives too challenging
      • Numerical Derivatives too slow
    • Derivative Free Optimization
FRaZ: Approach

• Why not use binary search?
  • It doesn’t work
  • The relationship between error bounds and compression ratios is not monotonic

• What can we use?
  • Optimization
    • Derivative Based Methods
      • Analytic Derivatives too challenging
      • Numerical Derivatives too slow
    • Derivative Free Optimization
FRaZ: Key Findings

- Tuning takes only 2x longer than an oracle in the feasible case
- Some targets are faster because more error bounds meet some targets
- How do we get there?
  - Parallelize by
    1. Field – run each field independently
    2. Timestep – try reusing prior timesteps configuration
    3. Error Bound Range – run ranges independently, stopping early if a solution is found
Automated Configuration Timeline

In Submission

FRaZ
- Bounding Compression Ratio

LibPressio-Opt
- Bounding User Metrics
- Performance Improvements

More Complex Apps
- Bounding Complex, Multi-faceted Metrics
Can we extend FRaZ to bound simple user metrics and improve performance?
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\vec{c} \in U} Q\left( d_{f,t}, \tilde{d}_{f,t}(\vec{c}; \vec{\theta}_c) \right)
\]
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

\[ \max_{\tilde{c} \in U} Q(d_{f,t}, \bar{d}_{f,t}(\tilde{c}; \theta_c); \theta_m) \]

- Compression Ratio
- Data for a Field and Timestep
- Error Bound Mode
- Error Bound
- Compression Function
- Allowed Error Bounds
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\vec{c} \in U} Q \left( d_{f,t}, \tilde{d}_{f,t} (\vec{c}; \theta_c); \theta_m \right)
\]
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\vec{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\vec{c}; \theta_c); \theta_m)
\]
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

$$\max_{\tilde{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\tilde{c}; \theta_c); \theta_m)$$

- User Error Bound
- Data for a Field and Timestep
- Error Bound Mode
- Non-fixed Compressor Settings
- Allowed Error Bounds
- Compression Function
- Fixed Metric Parameters
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

\[
\max_{\hat{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\hat{c}; \theta_c); \theta_m)
\]
LibPressio-Opt: Approach

Formulate compressor configuration as an optimization problem

$$\max_{\tilde{c} \in U} Q(d_{f,t}, \tilde{d}_{f,t}(\tilde{c}; \tilde{\theta}_c); \tilde{\theta}_m)$$

- User Error Bound
- Data for a Field and Timestep
- Compression Function
- Fixed Metric Parameters
- Non-fixed Compressor Settings
- Feasible Compressor Settings
- Compressor Fixed-Parameters
Illustration of Relationship among Notations

- Compression parameter space $\Omega$
- Uncompressed data buffer (i.e., original raw data)
- Decompressed data buffer

- User-analysis parameter space
- $\tilde{\theta}_c$
- $\theta_m$

- $\tilde{c}$

- $D$ (Dataset)

- $D'$ (Dataset)

- Timesteps per Dataset $(T_D, F_D)$

- Fields per Dataset
What about constraints on objectives?

For any, \( Q_i(df, t, \tilde{df}, t (\tilde{c}; \tilde{\theta}_c)) \) and \( \tau_i \)

We can construct:

\[
Q \left( df, t, \tilde{df}, t (\tilde{c}; \tilde{\theta}_c) \right) =
\begin{cases}
Q_0 \left( df, t, \tilde{df}, t (\tilde{c}; \tilde{\theta}_c) \right) & \text{if } \forall_i, Q_i \left( df, t, \tilde{df}, t (\tilde{c}; \tilde{\theta}_c) \right) \leq \tau_i \\
-\infty & , \text{otherwise}
\end{cases}
\]
What do we mean by “Metrics”?

• Requirements
  • “Sufficiently” Deterministic
  • Have a fixed number of Real valued inputs and outputs
  • Can be modeled as having a single objective
What do we mean by “Metrics”? 

- Requirements 
  - “Sufficiently” Deterministic 
  - Have a fixed number of Real valued inputs and outputs 
  - Can be modeled as having a single objective 

- Types 
  
  Real Time – Can only be computed at runtime (i.e. compression_time)
What do we mean by “Metrics”?

• Requirements
  • “Sufficiently” Deterministic
  • Have a fixed number of Real valued inputs and outputs
  • Can be modeled as having a single objective

• Types

Application – Needs a specific collection of buffers to compute (Anything “app” specific)
What do we mean by "Metrics"?

• Requirements
  • “Sufficiently” Deterministic
  • Have a fixed number of Real valued inputs and outputs
  • Can be modeled as having a single objective

• Types

Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)
What do we mean by “Metrics”?

- **Requirements**
  - “Sufficiently” Deterministic
  - Have a fixed number of Real valued inputs and outputs
  - Can be modeled as having a single objective

- **Types**
  
  Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)
  
  Single-Buffer – Computed from any single buffer
What do we mean by “Metrics”?  

- Requirements  
  - “Sufficiently” Deterministic  
  - Have a fixed number of Real valued inputs and outputs  
  - Can be modeled as having a single objective  

- Types  
  Multi-Buffer – Can be computed with any number of buffers (i.e. compression_ratio)  
  Single-Buffer – Computed from any single buffer
MGARD Quantity of Interest Mode

- **Requirements**
  - Q is a bounded linear functional
  - iff: $Q(\alpha x + \beta y) = \alpha Q(x) + \beta Q(y)$
  - $d_{f,t}$ represents a regular grid
    - This includes many simulations

- **Procedure**
  - Precompute scaling factor $\Upsilon_{L^s}(Q)$
  - Use bound $\Upsilon_{L^s}(Q) \| d_{f,t} - \tilde{d}_{f,t} \|_{L^s}$
  - Details in the paper cited below

Ainsworth, Mark; Tugluk, Ozan; Whitney, Ben; Klasky, Scott. "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities. 2019
VS. MGARD Quantity of Interest Mode

- Relative to MGARD-QOI mode
  - LibPressio-Opt+SZ is much faster for one-off tasks
  - Even if precomputation is not required, it can still be faster
VS. FRaZ

• Relative to FRaZ
  • Inter-iteration early termination
  • Multi-threaded searches
  • Embeddable
  • Supports user-defined objectives
  • Supports multiple input parameters
  • Extendable Search methods
Automated Configuration Timeline

1. FRaZ
   - Bounding Compression Ratio

2. LibPressio-Opt
   - Bounding User Metrics
   - Performance Improvements

3. More Complex Apps
   - Bounding Complex, Multi-faceted Metrics

Current Work
3 – More Complex Applications

Can we adapt LibPressio-Opt to bound complex, multi-faceted metrics that use multiple buffers such as Hardware/Hybrid Accelerated Cosmology Code (HACC)’s power spectrum?
Background

- HACC – ECP astrophysics particle application
- No compressors bound spectral error

Matter Fluctuation Power Spectra from Outer Rim and Q Continuum simulation at z=0.26. [3]
Approach

1. Implement the spectra as a LibPressio metric of type vector<double>
2. Explore metrics to compare spectra
3. Solve maximum compression ratio such that differences between spectra are acceptable
Future Work

• True Multi-Objective Compression
• Improve the search algorithm
• Better Compression task scheduling
Understanding the Effects of Compression on ML/AI
Lossy Compression for AI

What are the trade-offs for compressing training and testing data to save storage space?
Approach

• Use LibPressio External Metrics on training data to collect pareto-optimal points
  • External Metrics run scripts to collect data.
  • In this case: here a known-good AI based model
Key Findings

- Prediction-Based EBLC works best (SZ)
  - Even better than sampling!
  - Even on imbalanced datasets!
- Sometimes EBLC improves performance!
- Compress tabular data relatively by feature
Conclusion

• Error Bounded Lossy Compression has the potential to be transformative
  • Especially with an interface to unify, tools to configure, and tools to understand
Thank You!
Thank You!
Questions?

Approachable Error Bounded Lossy Compression

Robert Underwood
robertu@g.clemson.edu
Clemson University
October 7, 2020

My Curriculum Vita
Papers I Worked On


• A. Gok et al. “Metrics for the Preservation of the Error In Derivatives” 2020 (In Preparation)


• R. Underwood et al. “Machine Learning and AI with Error Bounded Lossy Compression” (In Preparation)
References

• Ainsworth, Mark; Tugluk, Ozan; Whitney, Ben; Klasky, Scott. "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities. 2019
• Di, Sheng and Cappello, Franck “Fast Error Bounded Lossy Compression with SZ” 2016
• Franck Cappello et al. “Use Cases of Lossy Compression for Floating Point Data in Scientific Datasets. 2018
• Lindstrom, Peter. “Fixed Rate Compressed Floating Point Arrays” 2012
• Whitney, Ben E. “Multilevel Techniques for Compression and Reduction of Scientific Data” 2018

• Machine Characteristics from respective websites accessed 17 September 2020
• Logos are the property of the respective institutions
Funding Acknowledgements

Thank you to Clemson Computing and Information Technology for funding my participation in SC20

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, Responsible for the planning and preparation of a capable Exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation’s Exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant No. 1619253 and 1910197. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory. This material is also based upon work supported by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists, Office of Science Graduate Student Research (SCGSR) program. The SCGSR program is administered by the Oak Ridge Institute for Science and Education (ORISE) for the DOE. ORISE is managed by ORAU under contract number DE-SC0014664. All opinions expressed in this paper are the authors and do not necessarily reflect the policies and views of DOE, ORAU, or ORISE.
Installing LibPressio

Prerequisites:
• LibPressio is currently only supported on Linux
• python>=3.4, tar, bzip, xz, gzip, clang/gcc, curl, git, bash, patch

Installation:
  git clone https://github.com/spack/spack
  git clone https://github.com/robertu94/spack_packages robertu94_packages
  source ./spack/share/spack/setup-env.sh
  spack compiler find
  spack repo add ./robertu94_packages
  spack install libpressio-tools ^libpressio+sz+zfp+lua
  spack load libpressio-tools
Overview of the structures in LibPressio
pressio – library control

- Query
  - Supported compressors
  - Version information
- Create compressor instances

```c
//get the compressor
struct pressio* library = pressio_instance();
struct pressio_compressor* sz = pressio_get_compressor(library, ← "sz");
```
pressio_options – key-value store

• map<string, option>
• One structure multiple uses:
  • Options – runtime settings
  • Configuration – compile time settings
  • Metrics Results – runtime observations
• Introspection
• Get/Set/Cast
• Validation of Settings

```cpp
// configure, validate, and assign the options
struct pressio_options* config = ...
pressio_options_set_integer(config, "sz:error_bound_mode", REL);
pressio_options_set_double(config, "sz:rel_err_bound", 0.01);
pressio_compressor_set_options(sz, config);
```
pressio_io

• Required Interfaces
  • read/write

• Optional Interfaces
  • configurable
    • {get,set,check}_options
    • get_configuration
  • {read,write}_many

//read in an input buffer
size_t dims[] = {500, 500, 100};
struct pressio_data* description = pressio_data_new_empty(pressio_float_dtype, 3, dims);
struct pressio_data* input_data = pressio_io_data_path_read(description, "CL0UDf48.bin.f32");
pressio_data – generic reference to data

- span<T>
- Query
  - type
  - size
  - values
  - owning/non-owning
- Extensible

```c
//create output buffers
struct pressio_data* compressed_data =
  pressio_data_new_empty(pressio_byte_dtype, 0, NULL);
struct pressio_data* decompressed_data =
  pressio_data_new_owing(pressio_float_dtype, 3, dims);
```
pressio_compressor

• Required Interfaces:
  • compress/decompress
  • version

• Optional Interfaces:
  • {compress,decompress}_many
  • configurable
  • get_metrics_impl

```c
//compress and decompress the data
pressio_compressor_compress(sz, input_data, compressed_data);
pRESSIO_COMPRESSOR_DECOMPRESS(sz, compressed_data, ...
                               ... decompressed_data);
```
pressio_metrics

• Required Interface
  • get_metrics_results

• Optional Interfaces
  • configurable
  • before_${compressor_fn}
  • after_${compressor_fn}
Using the Command Line

pressio \n  -t float -d 500 -d 500 -d 100 \n  -i $datasets/CLOUDf48.bin.f32 \n  -m size -m time \n  -M all \n  -o sz:abs_err_bound=1e-6 \n  -o sz:error_bound_mode_str="ABS" \n  sz

#use the command line
# the data is 32-bit float, dims=500x500x100
# the data is a “brick-of-floats” here
# collect size and time metrics
# print all collected metrics
# set SZ’s abs error bound to 1e-6
# set SZ’s error bound mode to ABS
# use SZ as the “root” compressor
Learn More about LibPressio

• Want to get Started?
  • Watch the "LibPressio Tutorial" on YouTube
  • Reference the extensive developer documentation
  • Check it out on Github

Get LibPressio