MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

MaRDIMark : *the MaRDI benchmarking framework and its instances*

A. S. Nayak, K.Lund, J. Saak, P. Benner

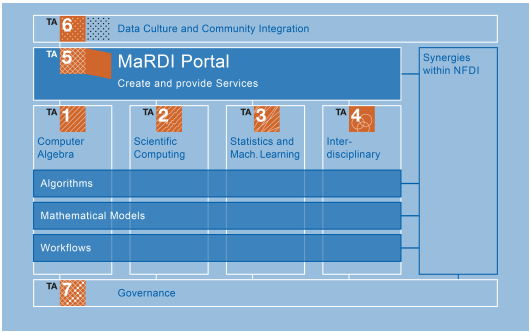
December 5, 2025

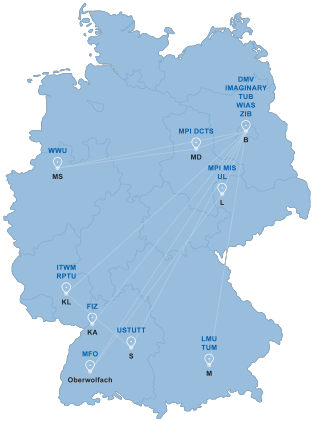
GAMM RSE/RDM Kickoff Meeting

Supported by:

MaRDI: Mathematical Research Data Initiative

- The German research data consortium for mathematics.
- 16 institutions and partners.
- 2021-2026.
- Supported by **National Research Data Infrastructure** (NFDI).





Outline

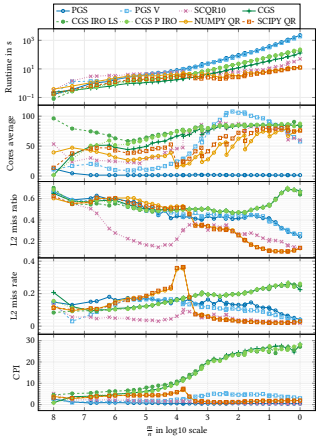
Introduction

Benchmarking : Introduction

- Benchmarking is a process of **comparing** the **performance** of a system or component using a set of **standard measures**.
  - Requires a **problem** definition with a set of **data** and a set of **algorithms**.
  - Requires a well-defined set of **metrics**.
- A **transparent** benchmarking process is essential for:
  - identify areas for improvement,
  - optimize efficiency and effectiveness,
  - ensure compliance with standards.

Universal Benchmarker

- The definition is **community-specific** and,
- depends heavily on the **context**.
- There is **no universal benchmarker**!

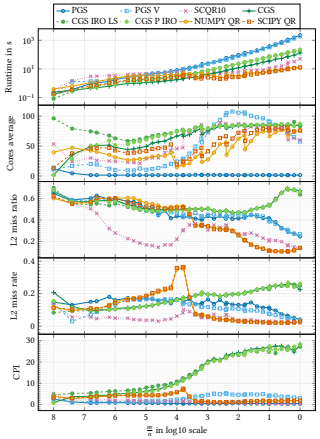


Benchmarking : Introduction

- Benchmarking is a process of **comparing** the **performance** of a system or component using a set of **standard measures**.
  - Requires a **problem** definition with a set of **data** and a set of **algorithms**.
  - Requires a well-defined set of **metrics**.
- A **transparent** benchmarking process is essential for:
  - identify areas for improvement,
  - optimize efficiency and effectiveness,
  - ensure compliance with standards.

Universal Benchmarker

- The definition is **community-specific** and,
- depends heavily on the **context**.
- There is **no universal benchmarker**!



Benchmarking : Problem Constituents

Database Curation

- Define benchmark instance and “concrete algorithm”
- Determine important searchable attributes and aggregate metadata
- Automate curation process to avoid human error
- Choose file-naming schemes and standards that are FAIR and conform to community traditions

Benchmark Framework

- Identify domain-specific aspects of each module
- Choose intuitive and informative performance measures
- Develop platform-independent interfaces

Community Engagement

- Ensure proper licensing
- Encourage researchers to contribute their data, provide feedback, and conform to standards
- KISS[1]: Reduce barriers to cooperation by providing workflows, GUIs, easy-to-follow guidelines, etc.

[1] Keep It Simple, Silly
https://w.wiki/JAn

Benchmarking : Tools

Computer Science

- google/benchmark
- catchorg/Catch2
- pytest-dev/pytest-benchmark
- …

Focus on testing and providing detailed computational **performance metrics** for algorithms. Popular due to their **easy-to-use** yet feature-rich API.

Distributed Computing

- FZJ-JSC/JUBE
- LLNL/benchpark
- GoogleCloudPlatform/PerfKitBenchmarker
- …

Features include **reproducible** complicated software configurations across distributed systems, **parametric runs**, and **comprehensive reporting**.

- **Reproducibility**: Consistency in results across multiple runs.
- **Scalability**: Handle different sizes of data and systems.
- **Fairness**: Unbiased comparison across different hardwares or software tools.
- **Verification**: Confirmable and verifiable results.
- **Ease of use**: Simple and intuitive.

Benchmarking : Tools

Computer Science

- google/benchmark
- catchorg/Catch2
- pytest-dev/pytest-benchmark
- …

Focus on testing and providing detailed computational **performance metrics** for algorithms. Popular due to their **easy-to-use** yet feature-rich API.

Distributed Computing

- FZJ-JSC/JUBE
- LLNL/benchpark
- GoogleCloudPlatform/PerfKitBenchmarker
- …

Features include **reproducible** complicated software configurations across distributed systems, **parametric runs**, and **comprehensive reporting**.

- **Reproducibility**: Consistency in results across multiple runs.
- **Scalability**: Handle different sizes of data and systems.
- **Fairness**: Unbiased comparison across different hardwares or software tools.
- **Verification**: Confirmable and verifiable results.
- **Ease of use**: Simple and intuitive.

Benchmarking : Tools

Computer Science

- google/benchmark
- catchorg/Catch2
- pytest-dev/pytest-benchmark
- …

Focus on testing and providing detailed computational **performance metrics** for algorithms. Popular due to their **easy-to-use** yet feature-rich API.

Distributed Computing

- FZJ-JSC/JUBE
- LLNL/benchpark
- GoogleCloudPlatform/PerfKitBenchmarker
- …

Features include **reproducible** complicated software configurations across distributed systems, **parametric runs**, and **comprehensive reporting**.

- **Reproducibility**: Consistency in results across multiple runs.
- **Scalability**: Handle different sizes of data and systems.
- **Fairness**: Unbiased comparison across different hardwares or software tools.
- **Verification**: Confirmable and verifiable results.
- **Ease of use**: Simple and intuitive.

The MarDIMark specification

MaRDIMark : A benchmarking specification

Examples

Contains datasets, parameters and problem descriptions of unique problem instance.

Algorithms

Includes code, descriptions of software and metadata for specifying concrete algorithms.

Driver

Implements interfaces to algorithms, orchestrates benchmark runs and collects raw data for results.

Analyzer

Performs post-processing and evaluates performance measures on raw result data.

Explorer

Generates visualizations and human-readable, platform-independent reports.

A generic, extensible benchmark framework specification

- Generic extensible toolkit specification
- Language agnostic interoperability
- FAIR comparison among different algorithm implementations
- Flexible performance measures
- Versatile visualizations of results

- Connect with Knowledge graphs
- Use open interfaces
- Confirmable workflows
- Integrate into MaRDI portal

MORB: A MarDIMark instance

MORB : Model Order Reduction Benchmarker

- A **MaRDIMark** implementation for the **Model Order Reduction** community.
- Serve as a template for other mathematical communities.
- Focus on Linear Time-Invariant, First-Order Systems (**LTI-FOS**).

$$E\,\dot{x}(t) = A\,x(t) + B\,u(t)$$
$$y(t) = C\,x(t) + D\,u(t)$$

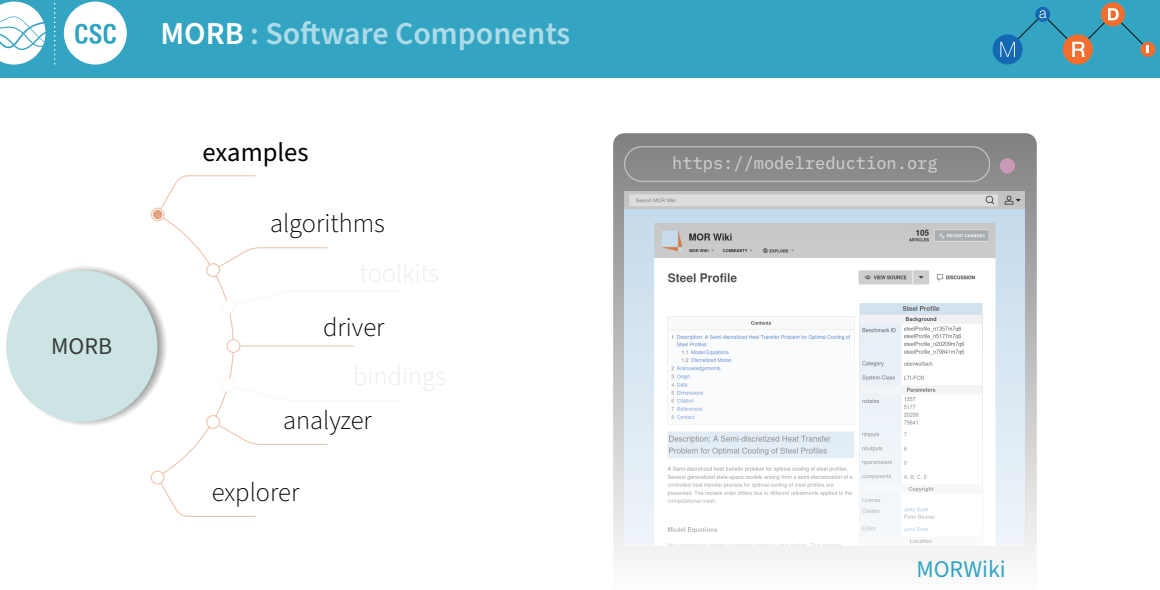
MOR

$$\widehat{E}\,\dot{\hat{x}}(t) = \widehat{A}\,\hat{x}(t) + \widehat{B}\,u(t)$$
$$\hat{y}(t) = \widehat{C}\,\hat{x}(t) + \widehat{D}\,u(t)$$

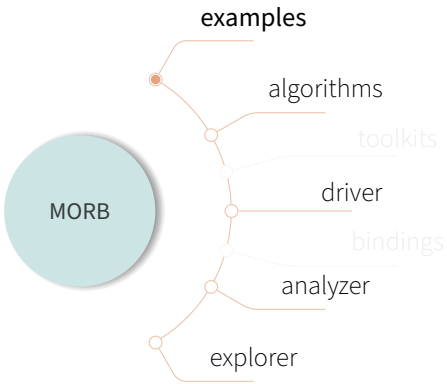
MORB : Design Challenges

- **Community-based** data entries and easy dataset fetching on demand.
- Ensuring dataset is tagged heavily and encoded **uniformly**.
- Easy **accessibility** to algorithm information.
- Distinguish unique implementation of an algorithm (“**concrete algorithm**”)
- Calling external software as “**black box**”-es and without unnecessary overhead.
- Implement subroutines that compute **measures** (e.g. error, speed etc.) efficiently, accurately, FAIR-ly.
- Provide a user-friendly interface for data **visualization** and analysis.
- **Platform-agnostic** and highly **configurable** interface with a low threshold to learn, but easy to extend.
- Workflow should be made **reproducible**, **reusable** and **replicable**.

MORB : Software Components



MORB : Software Components



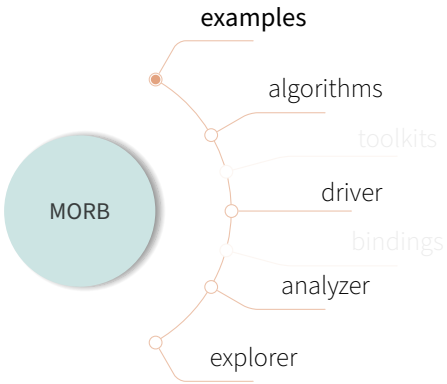
examples

- Matrix data checked in **LTISystem** class.
- Can retrieve datasets using `morb-fetch`.
- Sorts the dataset based on combination of matrices received.
- Can also be configured with custom datasets.
- (ongoing) connect to knowledge graphs.

`morb-fetch` `mardi4nfdi/morb-fetch`

- Easy dataset selection, download, extraction
- Flexible configuration options and dataset-caching
- Python package with easy installation
- Platform independent

MORB : Software Components



examples

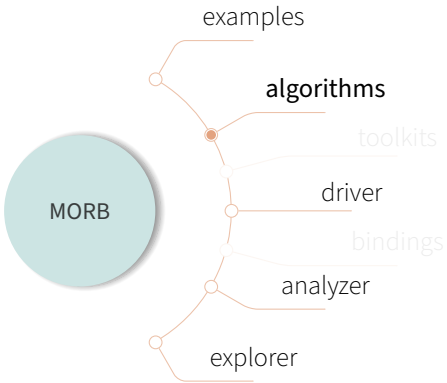
- Matrix data checked in **LTISystem** class.
- Can retrieve datasets using `morb-fetch`.
- Sorts the dataset based on combination of matrices received.
- Can also be configured with custom datasets.
- (ongoing) connect to knowledge graphs.

`morb-fetch`

`mardi4nfdi/morb-fetch`

- Easy dataset selection, download, extraction
- Flexible configuration options and dataset-caching
- Python package with easy installation
- Platform independent

MORB : Software Components



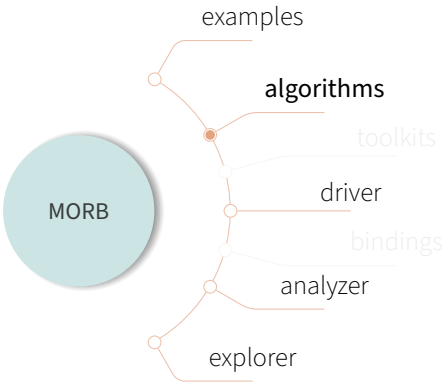
algorithms

- Currently only implements **balancing** methods.
  - Balanced Truncation (BT)
  - Positive-real BT
  - Stochastic BT
  - Frequency-limited BT
  - Time-limited BT
  - Frequency-weighted BT
  - Linear Quadratic Gaussian BT
- holds metadata pointing to a **MORWiki** entry.
- (extend) connect to knowledge graphs.

concrete algorithm

- Implementable algorithm with fully specified parameters.
- Stores identifiable information of a **toolkit**.
- Can `apply()` itself on an **example**.

MORB : Software Components



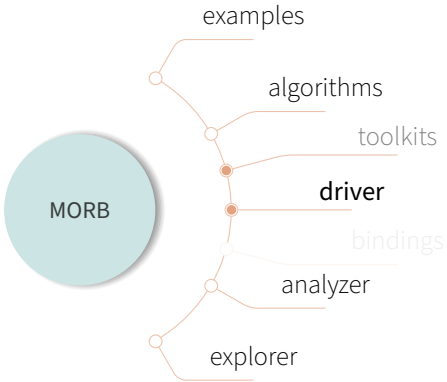
algorithms

- Currently only implements **balancing** methods.
  - Balanced Truncation (BT)
  - Positive-real BT
  - Stochastic BT
  - Frequency-limited BT
  - Time-limited BT
  - Frequency-weighted BT
  - Linear Quadratic Gaussian BT
- holds metadata pointing to a **MORWiki** entry.
- (extend) connect to knowledge graphs.

concrete algorithm

- Implementable algorithm with fully specified parameters.
- Stores identifiable information of a **toolkit**.
- Can `apply()` itself on an **example**.

MORB : Software Components



toolkits

- M-M.E.S.S
- MORLAB
- pyMOR
- Octave Control Toolbox
- MATLAB Control Toolbox
- …(user extensible)

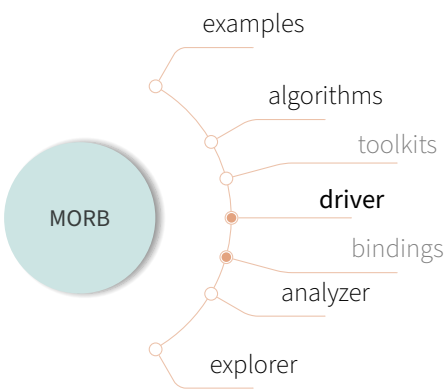
bindings

- Octave API
- MATLAB API
- Python API
- Julia API
- C++ API
- …(user extensible)

driver

- Parses the **problem** description and orchestrates it's execution.
- Interfaces to different **toolkits**.
- Provides **executors** which manages **bindings**.

MORB : Software Components



toolkits

- M-M.E.S.S
- MORLAB
- pyMOR
- Octave Control Toolbox
- MATLAB Control Toolbox
- …(user extensible)

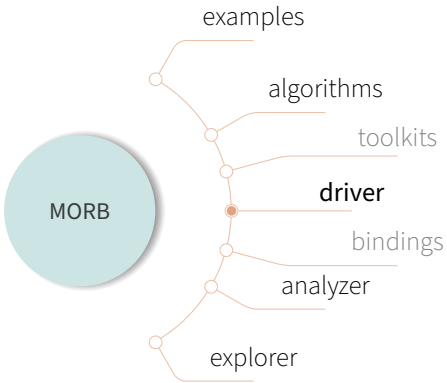
bindings

- Octave API
- MATLAB API
- Python API
- Julia API
- C++ API
- …(user extensible)

driver

- Parses the **problem** description and orchestrates it's execution.
- Interfaces to different **toolkits**.
- Provides **executors** which manages **bindings**.

MORB : Software Components



toolkits

- M-M.E.S.S
- MORLAB
- pyMOR
- Octave Control Toolbox
- MATLAB Control Toolbox
- …(user extensible)

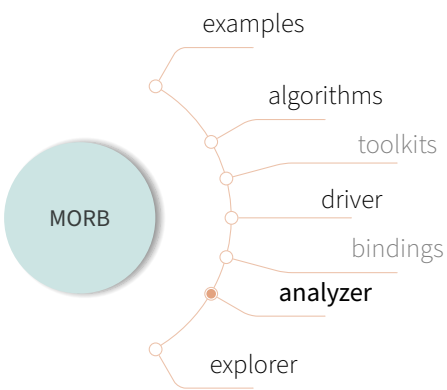
bindings

- Octave API
- MATLAB API
- Python API
- Julia API
- C++ API
- …(user extensible)

driver

- Parses the **problem** description and orchestrates it's execution.
- Interfaces to different **toolkits**.
- Provides **executors** which manages **bindings**.

MORB : Software Components



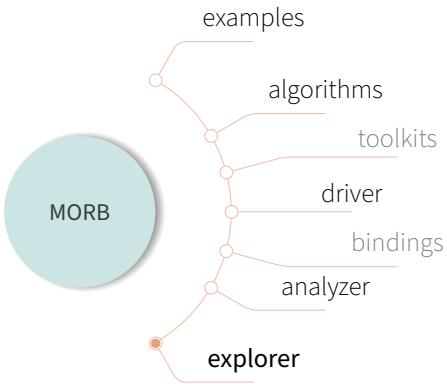
analyzer

- Defines proper **measures** for accuracy, performance and reliability of the algorithms.
  - Error Norms : $L^1, L^2, H^2, H^\infty, \ldots$
  - Bode Magnitudes
  - Singular values
  - Computation Runtimes
- (extend) upload to knowledge graph.

explorer

- Plots: Timings, Error plots.
- TEX Report: autogenerated with specifications, extensible templates.
- Tectonic-based fallback compiling. Easily configurable.
- PDF Report: Easily distributed and viewed.

MORB : Software Components



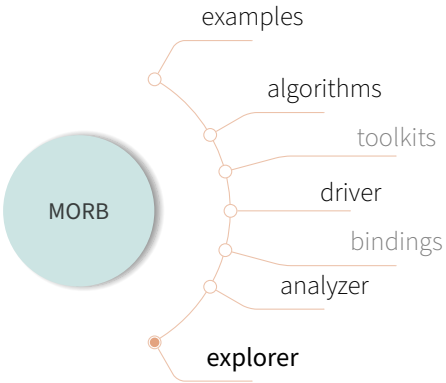
analyzer

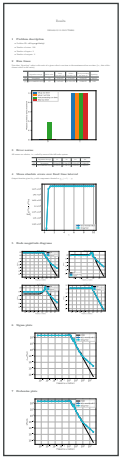
- Defines proper **measures** for accuracy, performance and reliability of the algorithms.
  - Error Norms : $L^1, L^2, H^2, H^\infty, \ldots$
  - Bode Magnitudes
  - Singular values
  - Computation Runtimes
- (extend) upload to knowledge graph.

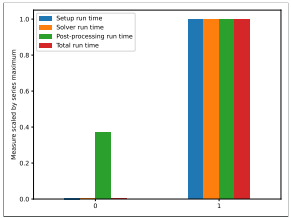
explorer

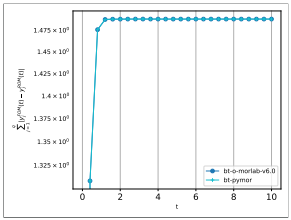
- Plots: Timings, Error plots.
- TEX Report: autogenerated with specifications, extensible templates.
- Tectonic-based fallback compiling. Easily configurable.
- PDF Report: Easily distributed and viewed.

MORB : Software Components









MORB : Interface


```
{
  "mor_problem": {
    "lti_system": {
      "id" : "steelProfile_n1357m7q6"
    },
    "algorithm": {
      "balanced_truncation": {
        "tolerance":1e-6,
        "max_order":100
      },
      "toolkit": {
        "morlab": {
          "version": "6.0.0",
          "interpreter": {
            "octave": {
              "version": "8.4.0"
            }
          }
        }
      }
    }
  }
}
```

```
$ morb driver run
```

JSON-file with Command Line Interface

```python
# Fetch a system by Example ID
from morb.examples import LTISystem
fom = LTISystem.from_id('steelProfile_n1357m7q6')

# List algorithms
from morb.toolkits import MORLAB
from morb.bindings import Octave
from morb.algorithms import BalancedTruncation
bt = BalancedTruncation(
  tolerance=1e-6,
  max_order=100,
  toolkit=MORLAB(
    version='6.0.0',
    interpreter=Octave(version='8.4.0')
  )
)

# Formulate a problem
from morb.driver import MORProblem, SerialExecutor
problem = MORProblem(fom, bt)

# Run benchmark
benchmark = SerialExecutor([problem])
results = benchmark.run()

# Compare with measures
from morb.analyzer import HInfErrorNorm
measure.HInfErrorNorm(fom, results)
```

Python-based easy-to-use interface

MORB : Configuration and Reproducibility

Configurable

- Cross-platform reproducible setup (with `uv.lock`) with caching of all fetched resources
- Environment variable overrides with persistent `YAML` configuration files
- Command-line interface (in development) ⇝ `MaRDIFlow`.

Confirmable

- Data is made verifiable using checksums (`SHA256` hashes)
- Verifiable workflows through proper typing and data models with validation (hashable and exportable to `JSON` schema)

Dependencies

- `Python 3.10+`
- `numpy`
- `polars`
- `platformdirs`
- `pydantic`
- `pooch`
- `uv`
- `matlab.engine`
- `tectonicx`

MORB : Configuration and Reproducibility

Configurable

- Cross-platform reproducible setup (with `uv.lock`) with caching of all fetched resources
- Environment variable overrides with persistent `YAML` configuration files
- Command-line interface (in development) ⇝ `MaRDIFlow`.

Confirmable

- Data is made verifiable using checksums (`SHA256` hashes)
- Verifiable workflows through proper typing and data models with validation (hashable and exportable to `JSON` schema)

Dependencies

- `Python 3.10+`
- `numpy`
- `polars`
- `platformdirs`
- `pydantic`
- `pooch`
- `uv`
- `matlab.engine`
- `tectonicx`

MORB : Configuration and Reproducibility

Configurable

- Cross-platform reproducible setup (with `uv.lock`) with caching of all fetched resources
- Environment variable overrides with persistent `YAML` configuration files
- Command-line interface (in development) ⇝ `MaRDIFlow`.

Confirmable

- Data is made verifiable using checksums (`SHA256` hashes)
- Verifiable workflows through proper typing and data models with validation (hashable and exportable to `JSON` schema)

Dependencies

- `Python 3.10+`
- `numpy`
- `polars`
- `platformdirs`
- `pydantic`
- `pooch`
- `uv`
- `matlab.engine`
- `tectonicx`

gasnetbench

- Implements interfaces to various comptutational models for gas networks.
- Current focus: accuracy of time discretizations.
- Collection of benchmark networks.

LASEr

- Currently, implements interfaces for orthogonal decomposition (QR).
- Hardware counters as performance indicators.
- Broad range of linear algebra backends.

Outlook

- Recommend algorithms and examples from and store results in knowledge graphs
- Bindings should involve open-interfaces: `MaRDI4NFDI/open-interfaces`
- Improve “confirmable workflows”
- Ensure synergetic potential with other communities and MaRDI projects
- Integrate into MaRDI portal
- Extend MORB specification to PH systems (planned)

Thank You!
Questions?

Reach out: mardi-ta2@listserv.uni-muenster.de