MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

# Scientific Computer-Based Experiments

**J. Fehr, J. Heiland, C. Himpe, S. Rave, J. Saak**

Computational Methods in Systems and Control Theory Group
Max Planck Institute Magdeburg

ICERM Semester Program
Model and Dimension Reduction in Uncertain and Dynamic Systems
2020–03–03

**Supported by:**

Federal Ministry
for Economic Affairs
and Energy

ICERM

# Why This Tutorial?

**We often present computational results, so:**

- Assure ourselves of correctness.
- Justify against scrutiny.
- Facilitate efficient research process.
- Ensure long-term validity.
- It's science!

# Disclaimer

- The following is not a strict set of rules.
- View it as a collection of best-practices.
- Adapt these ideas to your use-case.

# The Foundation

**Mathematical Software ...**

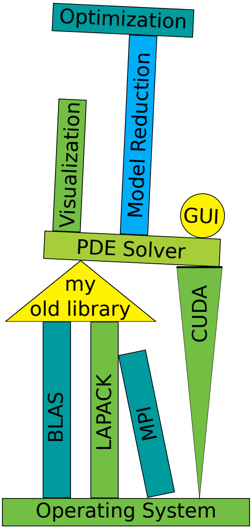
- ... has special responsibility,
- because it is the base layer,
- for most other computational sciences.

# The Engine

**Numerical Software ...**

- ... has its own additional challenges,
- foremostly finite-precision computations,
- and all this entails.

# Research Codes I




“Tower of Doom” (by: S. Rave)

# Research Codes II

"The Void" (by: C. Himpe)

# My Story

**Enter Christian:**

- Why are numerical experiments not available?
- Why can unproven properties be claimed?
- Why are methods not compared?
- Am I alone in pondering these questions?

# The Crew

Jörg Fehr
Uni Stuttgart

Jan Heiland
MPI Magdeburg

Christian Himpe
MPI Magdeburg

Stephan Rave
Uni Münster

Jens Saak
MPI Magdeburg

→ Together about one century of programming experience

## What is a CBEx?

- Any result obtained by a computer.
- No matter if it is:
- supporting or illustrative results,
- pointwise confirmation,
- or computational proof.

## What is a scientific CBEx?

- Any CBEx that *verifiably* does what its authors claim.

**Different types of codes:**

- Single purpose code
- Recyclable code

---

- Small projects
- Large projects

# CBEx Problems

**Sorted by increasing commonality:**

- Hardware not available
- Software stack not available
- Reporting not sufficient
- Archiving not stable
- Provisioning not sufficient
- **Lack of education**

# Previous Work

## A Selection:

1971 J.R. Rice, editor. **Mathematical Software**. ACM Monograph Series, Academic Press, 1971. doi:10.1016/C2013-0-11363-3

1979 H. Crowder, R.S. Dembo, J.M. Mulvey. **On Reporting Computational Experiments with Mathematical Software**. ACM Trans. Math. Software, 5(2): 193-203, 1979. doi:10.1145/355826.355833

1982 W.J. Cody. **Basic Concepts for Computational Software**. In: Problems and Methodologies in Mathematical Software Production, Lecture Notes in Computer Science, 142: 1-23, 1982. doi:10.1007/3-540-11603-6_1

1985 A.J. Laub. **Numerical Linear Algebra Aspects of Control Design Computations**. IEEE Trans. Automat. Control, 30(2): 97–108, 1985. doi:10.1109/TAC.1985.1103900

1997 R.F. Boisvert, editor. **Quality of Numerical Software**. IFIP Advances in Information and Communication Technology, Springer, 1997. doi:10.1007/978-1-5041-2940-4

2004 S. Van Huffel, V. Sima, A. Varga, S. Hammarling, F. Delebecque. **High-performance numerical software for control**. IEEE Control Systems Magazine 24(1): 60–76, 2004. doi:10.1109/MCS.2004.1272746

# Outline

- **Part I:** Replicability, Reproducibility, Reusability

  Based on [Fehr,Heiland,H.,Saak'16]

- **Part II:** Low-Hanging Files

- **Part III:** Sustainable Scientific Software

# The Three Rs

1. **R**eplicability
2. **R**eproducibility
3. **R**eusability

Each **R** has:

- Minimal requirements
- Optional recommendations

# Replicability

**Replicability** (aka Repeatability):

- **You** are able
- to **repeat**
- **your experiment**
- on **your computer**.

# Minimal Replicability Requirement

**Basic Documentation:**

- Recipe to obtain (numerical) results
- Recipe for post-processing of data
- Recipe for creating visualizations

**Automation and Testing:**

- Machine-readable recipes
- For example (shell) scripts
- Sanity tests

# Reproducibility

**Reproducibility:**

- **Somebody** (not you) is able
- to **repeat**
- **your experiment**
- on **their computer**.

# Minimal Reproducibility Requirement

**Detailed Documentation:**

- Environment description
- Versions of system and dependencies
- Building instructions (if applicable)

# Optional Reproducibility Recommendation

**Availability:**

- From a location with (long-term) storage purpose
- Storage is not bound to author
- Some identifier is provided

# Reusability

**Reusability:**

- **Somebody** is able
- to **use your experiment**
- on **their computer**
- for **something else**.

# Minimal Reusability Requirement

**Accessibility:**

- Availability
- Remote service required
- Binaries available (if applicable)

# Optional Reusability Recommendation

**Modularity, Software Management and Licensing:**

- Modular design
- Project management tools and resources
- License considerations

# Outline

- **Part I:** Replicability, Reproducibility, Reusability
- **Part II:** Low-Hanging Files
  a. Practical ideas for the paper
  b. Practical ideas for the code
- **Part III:** Sustainable Scientific Software

# Compute Environment

**Useful Minimal Information** (MATLAB, Octave, Python, R, Julia):

- Runtime interpreter name and version
- Operating system name, version and architecture / word-width
- *Processor name and exact identifier*
- Required amount of random access memory
- BLAS / LAPACK library implementation name and version

# Timing Results

**Pitfalls:**

- CPU time vs wall time
- Parallelization (implicit / explicit)
- Efficient memory access (NUMA)
- Overhead (actual compute-time)
- Statistics (i.e. means of repeated runs)

# Model Reduction Specific I

**Red Flags:**

- Not comparing to standard methods
  → Instead of POD using some obscure method
- Not justifying selected reduced orders
  → Competing methods may be as good at just one order higher
- Not using benchmarks
  → Testing solely on a system only you have access to
- Not describing free parameter choices
  → For example regularization weights
- Selective error quantification
  → Disregarding i.e. discretization error

# Model Reduction Specific II

**Deep Red Flags:**

- Not explaining or justifying measure of comparison
  → "Thus our method is the most efficient ..."
- Not using state-of-the art implementations
  → Your own implementation of a fourty year old pseudo-code
- Committing inverse crime / model reduction crime
  → Testing on the parameters you trained on
- Pessimizing competing methods
  → Wall-times of your parallel method versus a serial competitor
- Unfair comparisons

# Code Availability Section

**Numerical Results**

...

**Code Availability Section**

The source code of the implementations used to compute the presented results can be obtained from:

`https://my.stable.url`

and is authored by: X. Y., A. B.

(if available use supplemental material!)

# Outline

- **Part I:** Replicability, Reproducibility, Reusability
- **Part II:** Low-Hanging Files
  a. Practical ideas for the paper
  b. Practical ideas for the code
- **Part III:** Sustainable Scientific Software

# ALL CAPS

**Where to start:**

- Certain text files
- with quasi standard names
- and contents
- are established.

**Plain Text Files:**

- ASCII letters only (if possible)
- UTF-8 encoding
- Suitable for version control
- Can be decorated with markdown / commonmark.

# README

**Read this first:**

- Executive summary, functionality, basic information
- How to build / install / run / use
- Authors and contributors
- Breadcrumbs (website, papers, etc.)
- Table of contents (where to find documentation, etc.)

# RUNME

**Run this first:**

- This is an executable file.
- Typically a shell script.
- It computes the results,
- for example, plots used in a manuscript.
- It may need to be specialized (`.linux`, `.win`).

# CITATION

**How to Cite:**

- Sample citation
- BibTeX entry
- Citation guidelines
- Introduced by `R`
- Readable by `Octave`

# CODE

**Code Meta-Data**:

- Machine-readable and human-readable Key-Value Pairs
- Core information
- Format: `.ini` or `.json`
- Useful for discoverability

**Typical Keys:**

- name, shortname
- version, release-date
- id, id-type
- authors, ORCIDs
- topic, type
- license, license-type
- repository, repository-type
- languages, versions
- dependencies, versions
- systems, versions
- website
- keywords

# And Others ...

| | |
|---|---|
| `AUTHORS` | Who wrote it |
| `LICENSE` | The license text |
| `INSTALL` | How to install |
| `CHANGELOG` | What changed |
| `DEPENDENCIES` | What are the dependencies |
| `VERSION` | The version number |
| `TODO` | Open problems |
| `FAQ` | Frequently Asked Questions |
| | ... |

# Tag it, Bag it

**Source Code Header Info:**

- Software project or associated paper
- Authors (and contributors)
- Version of the code / project
- License of the file contents
- Summary of content (in one sentence)

# Outline

- **Part I:** Replicability, Reproducibility, Reusability
- **Part II:** Low-Hanging Files
- **Part III:** Sustainable Scientific Software
  - Based on [Fehr,H.,Rave,Saak'19]

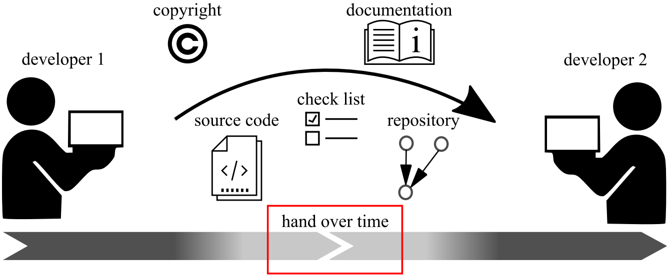
**Instead of:**

- Standing on the shoulders of giants,
- we are reinventing the wheel.

# Project Hand-Over

**Critical Situations:**

- Graduation (of a PhD or Master student)
- Changes (in the development team)
- Change (in the project lead)



# Project Types

1. Small Projects ("single developer")
2. Large Projects ("multiple developers")

Both have:

- Minimal requirements
- Optional recommendations
- Large projects imply small project requirements

# Small Project Requirements

- **Code Availability**
  - Hardware, source code, configuration, data
- **Code Ownership**
  - Owner, contributor, copyright-holder, third-party rights
- **Execution Environment**
  - Operating system, compiler, interpreter, dependencies
- **Working Example**
  - Sanity test, basic demo, `RUNME`
- **Minimal Documentation**
  - Functionality, state, algorithms, publications, limitations

# Small Project Recommendations

- **Public Release**
  - If not possible, note in `README`
- **Version Control**
  - Backup, history, collaboration, tagging
- **Basic Code Clean Up**
  - Magic numbers, dead code, hard-coded path
- **Reproducible Execution Environment**
  - Virtual machine with software stack, step-by-step guide
- **Integration into Larger Project**
  - Modularity, interfaces, contribution guidelines

# Large Project Requirements

- **Software License**
  - Formal agreement, open-source, implications,
- **Code Ownership of Contributions**
  - Copyright transfer
- **Access to Project Resources**
  - "Bus factor"
- **Management of Development Branches**
  - Feature / developer branches
- **Stable Main Branch**
  - Reliability

# Large Project Recommendations

- **Code Maintainability**
  - Continuous integration, continuous benchmarking
- **Changelog**
  - Project history
- **Code of Conduct**
  - Rules for project hand-over
- **Contribution Policy**
  - Legal status of contributions

# Bonus Tip

**Being Reviewer #3:**

- Require documentation of experiments.
- Require to see and run the source code.
- Suggest to make source code public.

# Materials I

- J. Fehr, J. Heiland, C. H., J. Saak. **Best Practices for Replicability, Reproducibility and Reusability of Computer-Based Experiments Exemplified by Model Reduction Software**. AIMS Mathematics 1(3): 261–281, 2016. `doi:bsb2`
- J. Fehr, C. H., S. Rave, J. Saak. **Sustainable Research Software Hand-Over**. arXiv, cs.GL: 1909.09469, 2019. `https://arxiv.org/abs/1909.09469`
- and references therein.

# Materials II

- R.J. LeVeque. **Top ten reasons to not share your code (and why you should anyway)**. SIAM News, 46, 2013. https://staff.washington.edu/rjl/pubs/topten/topten.pdf
- V. Stodden. **Implementing reproducibility in computational science**. SIAM Annual Meeting, 2016. https://www.pathlms.com/siam/courses/3028/sections/4128
- D. Procida. **What nobody tells you about documentation**. PyCon Australia, 2017. https://youtu.be/t4vKPhjcMZg

# MORwiki

**About the Model Order Reduction Wiki:**

- **A**ccessible Algorithms
- **B**enchmark Problems
- **C**ommunity Software

http://modelreduction.org

**Also on the MORwiki:**

- Community Calendar
- BibTeX Database (`mor.bib`)
- List of Introductory Works

# Summary

## Make your ...

- ... CBEx replicable, reproducible, reusable.
- ... code repository contain standard files.
- ... scientific software sustainable.

https://himpe.science

**Acknowledgements:**
Supported by the German Federal Ministry for Economic Affairs and Energy, in the joint project: "**MathEnergy** – Mathematical Key Technologies for Evolving Energy Grids", sub-project: Model Order Reduction (Grant number: 0324019**B**).

And by the National Science Foundation under Grant number DMS-1439786 while the author was in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the "**Model and dimension reduction in uncertain and dynamic systems**" semester program.