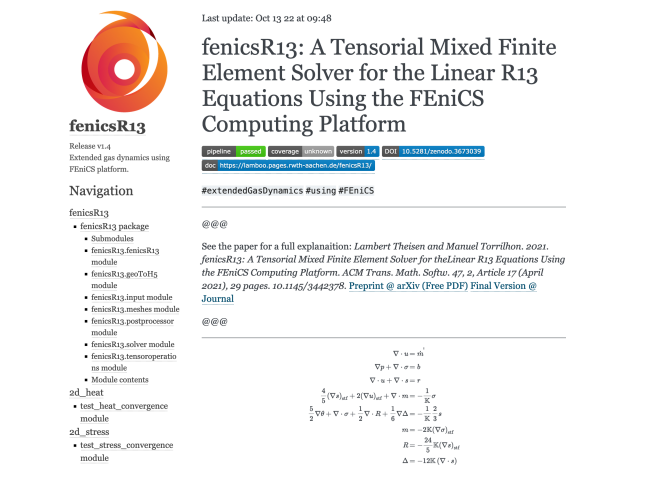
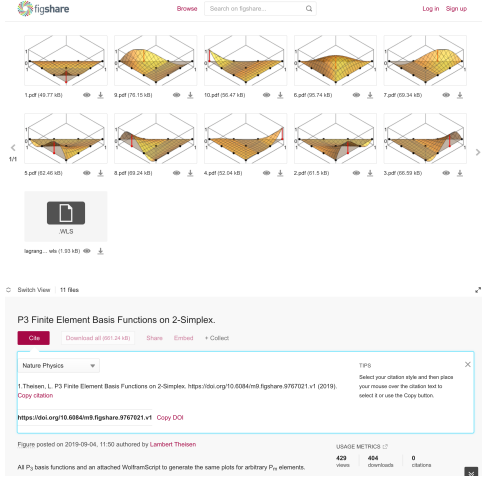




Publishing Reproducible Numerics: A Student's Perspective

RDM Workshop @ SFB 1481: Sparsity and Singular Structures

Lambert Theisen
NMH/IANS @ University of Stuttgart

Aachen, 05 September 2023

University of Stuttgart
Germany
Applied and Computational Mathematics
RWTH AACHEN UNIVERSITY

Outline

Outline

Motivation

Status Quo: Many use software for research[1]

Relevant Situations:

- You are given a paper as starting point for your project. First step: Understand and reproduce (my M.Sc. thesis)
- You want to continue an old project with a new idea (also as PI)
- You want to change an experiment later (for the PhD thesis?!)

[1] [Flemisch Gläser '23] [2] [Baker '16']

Motivation

Status Quo: Many use software for research[1]

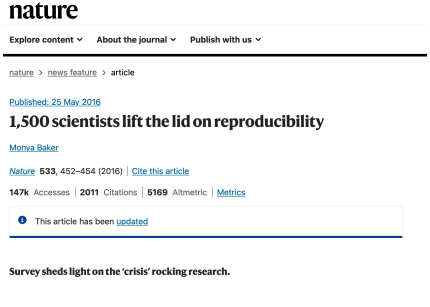
Relevant Situations:

- You are given a paper as starting point for your project. First step: Understand and reproduce (my M.Sc. thesis)
- You want to continue an old project with a new idea (also as PI)
- You want to change an experiment later (for the PhD thesis?!)

Problem: Reproducibility hard in practise[2]

- >70% failed to reproduce others
- >50% failed to reproduce **own** experiments -,-

⇒ We need reproducability strategies! But: No lab, only "computer"...



nature

Explore content ~ About the journal ~ Publish with us ~

nature > news feature > article

Published: 25 May 2016

1,500 scientists lift the lid on reproducibility

Monya Baker

Nature **533**, 452–454 (2016) | Cite this article

147k Accesses | **2011** Citations | **5169** Altmetric | Metrics

This article has been updated

Survey sheds light on the 'crisis' rocking research.

More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments. Those are some of the telling figures that emerged from *Nature*'s survey of 1,576 researchers who took a brief online questionnaire on reproducibility in research.

[1] [Flemisch Gläser '23] [2] [Baker '16']

Outline

What Does "Reproducible Numerics" Mean?

Definition:

Reproducing the exact numerical outcomes using the same setup.

Why?

- Establishes trust and ensures accuracy
- Building on previous work: standing on the shoulders of giants (avoid reinventing the wheel)

Typical TOCs

1. Introduction
2. The Gross–Pitaevskii equation and gradient flows
3. Adaptive gradient flow finite element discretization
4. Numerical Experiments
5. Conclusions

References

1 Introduction
2 Factorization and Homogenization of the Model Problem
3 Spatial Discretization and Iterative Eigensolvers
4 Numerical Experiments
5 Conclusion

References

1. Introduction
2. Unfitted DG and Trefftz DG Methods With Exact Geometry
3. Stabilisation Techniques
4. Error Analysis
5. Implementational Aspects
6. Numerical Examples

Data availability statement

References

What Does "Reproducible Numerics" Mean?

Definition:
Reproducing the exact numerical outcomes using the same setup.

Why?

- Establishes trust and ensures accuracy
- Building on previous work: standing on the shoulders of giants (avoid reinventing the wheel)

What can go wrong?

- Scripts not available, only method "description"
- Missing information, e.g. parameters $N, h, \eta_{\text{learn}}$, etc
- Unknown software versions, e.g. julia `v1.9.3` vs. julia `v1.6.7 LTS`
- Missing datasets (`mesh.msh`, `training-images.zip`, etc)
- Use of proprietary or unpublished, in-house software
- Other differences (Windows, macOS, Linux, etc) (M1, x64, etc)

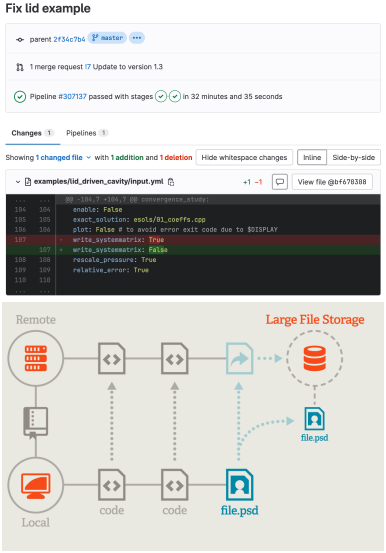
Typical TOCs

Outline

Reproducable Progress (with Git)

Version control with git

- **Problem**: `paper-v1.tex` → `paper-v3-final.tex` → `paper-v3-FINALLL.tex`
- Solution: Track only differences
- Workflow: `git add test.py` → `git commit -m "Add convergence test"` → `git push`
- Allows to go back in time, also allows for asynchronous collaboration
- Nowdays: DevOps platform to track issues, run automated tests, store dependencies, create roadmaps, etc
- Git LFS: For large binary files track are hard to "diff" (`mesh.msh`, `tomography.dat`, etc)



Reproducable Environments (with Docker)

Problem: People have different setups (student has macOS+python3.11 but PI has Windows+python3.7)

$\Rightarrow$ First step: Specify all dependencies

- e.g. `requirements.txt`, `setup.py` for Python
- e.g. `package.yml` and `manifest.yml` for Julia

But what about external deps (gmsh, TensorFlow, FEniCS, ...) or other cfg?

Reproducable Environments (with Docker)

Problem: People have different setups (student has macOS+python3.11 but PI has Windows+python3.7)

$\Rightarrow$ First step: Specify all dependencies

- e.g. `requirements.txt`, `setup.py` for Python
- e.g. `package.yml` and `manifest.yml` for Julia

But what about external deps (gmsh, TensorFlow, FEniCS, ...) or other cfg?

Virtualization with docker:

- Create common "Cooking recipe" ($\Leftrightarrow$ `Dockerfile`): Use *this*, install *that* ...
- Workflow: `docker build -tag myimg .` $\rightarrow$ `docker run -i -t myimg /bin/bash` to open a shell inside container
- IDE Coupling: Possible in VSCode
- Automation: Can be used for testing of code!

Example `Dockerfile`:

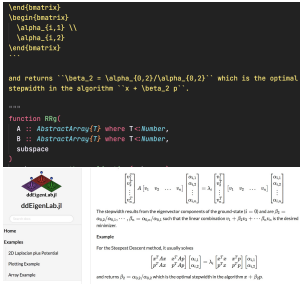
Dockerfile

```
# Use FEniCS base image
FROM quay.io/fenicsproject/stable:2019.1.0.r3

# Descriptions
LABEL maintainer="Lambert Theisen <lambert.theisen@rwth-aachen.de>"
LABEL description="Linearized R13 Equations Solver Environment"

# Specify software versions
ENV GMSH_VERSION 4.4.0

# Download Install Gmsh SDK with dependecies from Github's dolfinx Dockerfile
RUN export DEBIAN_FRONTEND=noninteractive && \
    apt-get -qq update && \
    # apt-get -yq --with-new-pkgs -o Dpkg::Options::="--force-confold" upgrade &&
    apt-get -y install \
        libglu1 \
        libxcursor-dev \
        libxinerama1 && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
RUN cd /usr/local && \
    wget -nc http://gmsh.info/bin/Linux/gmsh-${GMSH_VERSION}-Linux64-sdk.tgz && \
    tar -xf gmsh-${GMSH_VERSION}-Linux64-sdk.tgz
ENV PATH=/usr/local/gmsh-${GMSH_VERSION}-Linux64-sdk/bin:$PATH

# Install additional programs
RUN \
    apt-get update && \
    apt-get install -y \
    numdiff \
    htop \
    imagemagick

# Install any needed packages specified in requirements.txt
# RUN pip install --trusted-host pypi.python.org -r requirements.txt
WORKDIR /fenicsR13
ADD ./requirements.txt /fenicsR13/requirements.txt
RUN pip install -r /fenicsR13/requirements.txt
```

Reproducable Workflows (with Gitlab/-hub)

Explain usage: `README.md` for installation/execution
Documentation (automated)

- e.g. `Sphinx`, `Documenter.jl`, `doxygen`, ...
- Transform code comments (docstrings) to documentation, ideally host via Git pages



Reproducible Workflows (with Gitlab/-hub)

Explain usage: `README.md` for installation/execution
Documentation (automated)

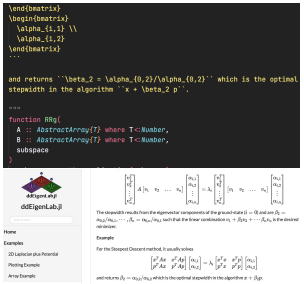
- e.g. `Sphinx`, `Documenter.jl`, `doxygen`, ...
- Transform code comments (docstrings) to documentation, ideally host via Git pages

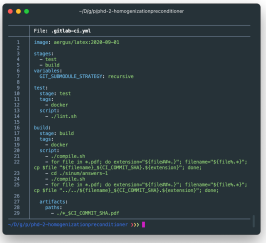
Testing with GitLab (RWTH instance) or GitHub

- CI/CD: Function from 6 month ago, still work?

⇒ Write tests for your methods

- Ideally: Also test the numerical examples (e.g. `numdiff` to captured output)
- **A good team**: Gitlab/-hub with automated tests. Every commit triggers the tests to run. Specifications via, e.g., `.gitlab-ci.yml` or `.github/workflows/[name].yml`
- Useful: Automatic LaTeX compilation (these slides)





Outline

Archiving Code for Publication (with Zenodo)

Setup: Code and paper ready for publication

Problem: Papers have a very long "lifecycle" but internet moves fast (Website links might change, people move)

[1] [Uekerman '23]

Archiving Code for Publication (with Zenodo)

Setup: Code and paper ready for publication
Problem: Papers have a very long "lifecycle" but internet moves fast (Website links might change, people move)

Digital Object Identifier (**DOI**) System:

- Organized by DOI Foundation, ISO standard, managed by registration agencies, `prefix/suffix`[1]
- Example: `10.1137/21M1456005`
- Resolver: `doi.org/10.1137/21M1456005` links to "correct" location

[1] [Uekerman '23]

Archiving Code for Publication (with Zenodo)

Setup: Code and paper ready for publication
Problem: Papers have a very long "lifecycle" but internet moves fast (Website links might change, people move)

Digital Object Identifier (**DOI**) System:

- Organized by DOI Foundation, ISO standard, managed by registration agencies, `prefix/suffix`[1]
- Example: `10.1137/21M1456005`
- Resolver: `doi.org/10.1137/21M1456005` links to "correct" location

Archiving using zenodo (or RWTH Publications in AC, DaRUS in STU)

- From CERN, creates DOI (citeable, unchangeable)
- Create versions (v1.0, v1.1, etc) of dataset/software
- Strategy: Git tag, download tarball, upload, cite DOI

[1] [Uekerman '23]

Archiving Code for Publication (with Zenodo)

Setup: Code and paper ready for publication

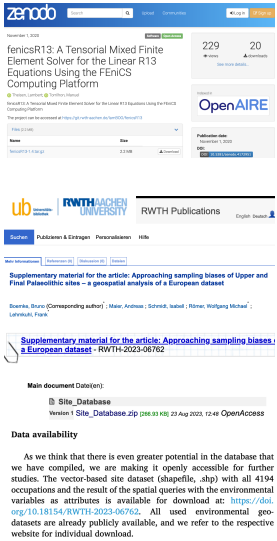
Problem: Papers have a very long "lifecycle" but internet moves fast (Website links might change, people move)

Digital Object Identifier (**DOI**) System:

- Organized by DOI Foundation, ISO standard, managed by registration agencies, `prefix/suffix`[1]
- Example: `10.1137/21M1456005`
- Resolver: `doi.org/10.1137/21M1456005` links to "correct" location

Archiving using zenodo (or **RWTH Publications** in AC, DaRUS in STU)

- From CERN, creates DOI (citeable, unchangeable)
- Create versions (v1.0, v1.1, etc) of dataset/software
- Strategy: Git tag, download tarball, upload, cite DOI



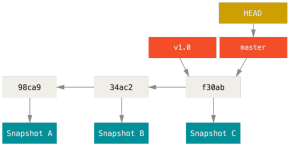
[1] [Uekerman '23]

Outline

Conclusion

Reproducable . . . progress with git

- Backup, collaboration, DevOps platform



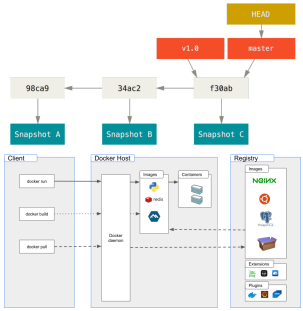
Conclusion

Reproducable . . . progress with git

- Backup, collaboration, DevOps platform

. . . environments with docker

- Specify all dependencies, create “cooking recipe”
- Hardware/software independence



Conclusion

Reproducable . . . progress with git

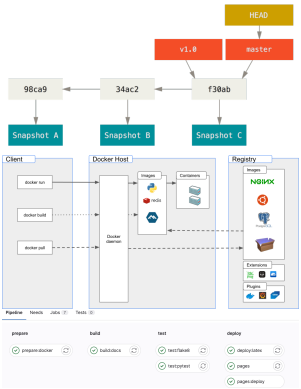
- Backup, collaboration, DevOps platform

. . . environments with docker

- Specify all dependencies, create “cooking recipe”
- Hardware/software independence

. . . workflows with GitLab or GitHub

- CI/CD automation (docu, testing, hosting, . . .)
- Confidence by automatic testing



Conclusion

Reproducable . . . progress with git

- Backup, collaboration, DevOps platform

. . . environments with docker

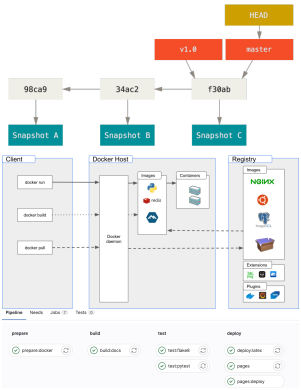
- Specify all dependencies, create "cooking recipe"
- Hardware/software independence

. . . workflows with GitLab **or** GitHub

- CI/CD automation (docu, testing, hosting, . . .)
- Confidence by automatic testing

Permanent archiving using zenodo

- Get permanent DOI identifier (citeable)



6. Numerical Examples

The previously described methods are implemented using `NGSTrefftz` [53] and `ngsxfem` [54], add-on packages to the finite element library `NGSolve/Netgen` [50,51]. The python scripts implementing the methods discussed in this paper and the full numerical results presented below are freely available in the zenodo repository [24]. The stabilisation parameters for the interior penalty and the Nitsche method are fixed in all examples to $\beta = 10k^2$ and the ghost penalty scaling to $\gamma = 0.01$ if not mentioned otherwise. For the solution of linear systems arising in the numerical examples, we used sparse direct solvers.

[24] F. Heimann, C. Lehrenfeld, P. Stocker, and H. von Wahl. Unfitted Trefftz discontinuous Galerkin methods for elliptic boundary value problems - Reproduction scripts, doi: 10.5281/zenodo.8020304, 2022.

Conclusion

Reproducable . . . progress with git

- Backup, collaboration, DevOps platform

. . . environments with docker

- Specify all dependencies, create "cooking recipe"
- Hardware/software independence

. . . workflows with GitLab **or** GitHub

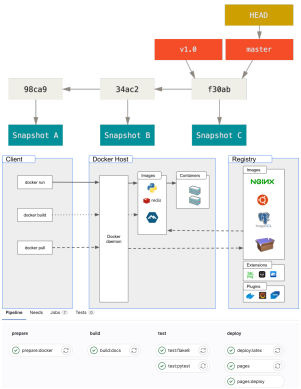
- CI/CD automation (docu, testing, hosting, . . .)
- Confidence by automatic testing

Permanent archiving using zenodo

- Get permanent DOI identifier (citeable)

Other aspects

- Interactivity (Jupyter notebooks), . . .



6. Numerical Examples

The previously described methods are implemented using `NGSTrefftz` [53] and `ngsxfem` [54], add-on packages to the finite element library `NGSolve/Netgen` [50,51]. The python scripts implementing the methods discussed in this paper and the full numerical results presented below are freely available in the zenodo repository [24]. The stabilisation parameters for the interior penalty and the Nitsche method are fixed in all examples to $\beta = 10k^2$ and the ghost penalty scaling to $\gamma = 0.01$ if not mentioned otherwise. For the solution of linear systems arising in the numerical examples, we used sparse direct solvers.

[24] F. Heimann, C. Lehrenfeld, P. Stocker, and H. von Wahl. Unfitted Trefftz discontinuous Galerkin methods for elliptic boundary value problems - Reproduction scripts, doi: 10.5281/zenodo.8020304, 2022.

References I

[1] M. Baker. "1,500 Scientists Lift the Lid on Reproducibility". In: *Nature* 533.7604 (2016), pp. 452–454. (Visited on 09/04/2023).

[2] B. Flemisch and D. Gläser. *Sustainable-Simulation-Software / Course-Material · GitLab*. https://gitlab.com/sustainable-simulation-software/course-material. 2023. (Visited on 09/04/2023).

[3] B. Uekermann. *Simulation-Software-Engineering/Lecture-Material: Material for the Simulation Software Engineering Lecture*. https://github.com/Simulation-Software-Engineering/Lecture-Material/tree/main. (Visited on 09/04/2023).

Questions?


lambert.theisen@ians.uni-stuttgart.de

lambert.theisen@rwth-aachen.de

@lamBOOO

https://thsn.dev


NMH @ Institute of Applied Analysis and Numerical Simulation

University of Stuttgart

German Research Foundation (DFG) project 411724963

Thank You.