RDM in practice:

Best practices for (small) computational projects

Oliver Melchert[1,2]

[1]Leibniz Universität Hannover, Institute of Quantum Optics
[2]Leibniz Universität Hannover, Cluster of Excellence PhoenixD

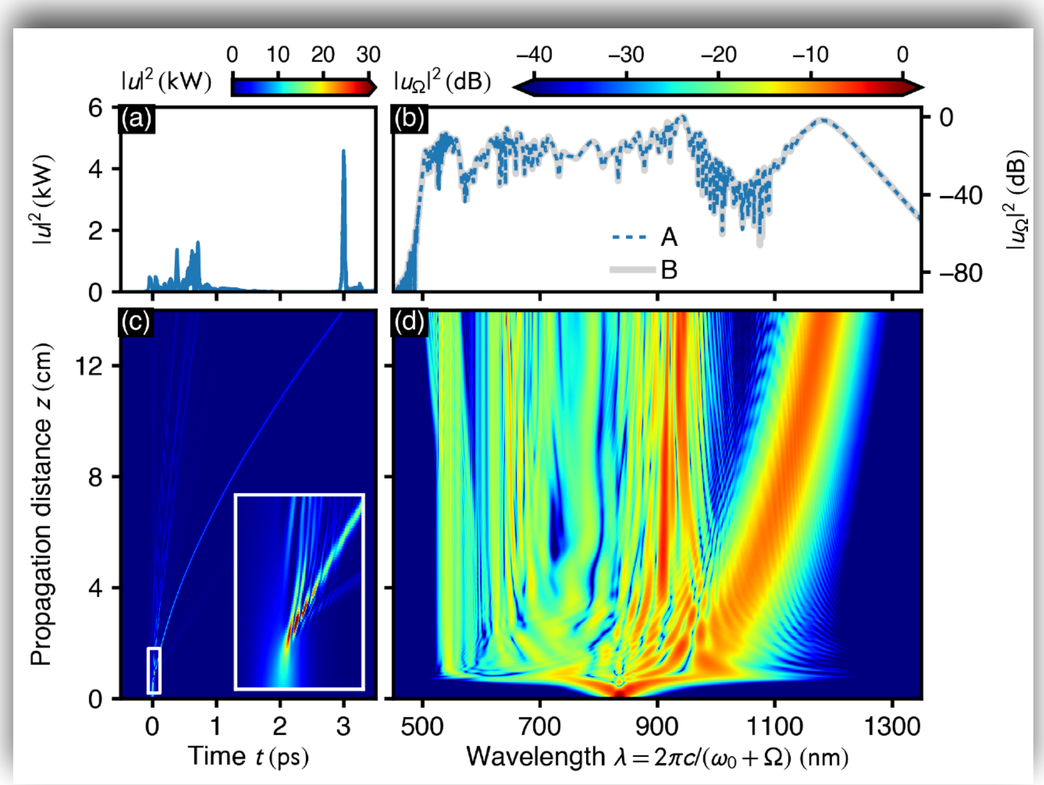
(EXC 2122, projectID 390833453)

Introduction / Outline

- **Introduction**
  - 15y of experience with computational projects
  - Researcher in nonlinear optics
  - Modelling + code development + simulation & analysis

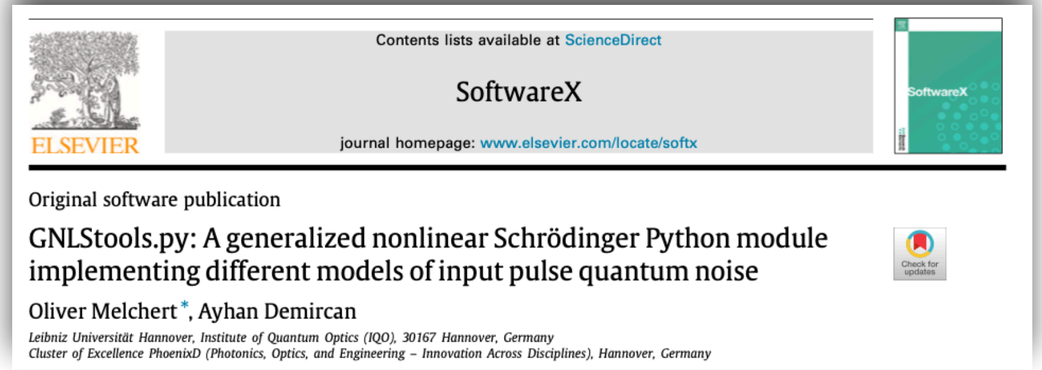
$$i\partial_z \mathcal{E}_\omega + \beta(\omega)\mathcal{E}_\omega + n_2 \frac{\omega}{c}\left((1-f_R)\,|\mathcal{E}|^2\mathcal{E} + f_R\,\mathcal{E}\mathcal{I}_R\right)_{\omega>0} = 0$$

$$\mathcal{I}_R = \sum_\omega h(\omega)\left(|\mathcal{E}|^2\right)_\omega e^{-i\omega t},\ h(\omega) = \frac{\tau_1^{-2}+\tau_2^{-2}}{\tau_1^{-2}-(\omega+i\tau_2^{-1})^2}$$



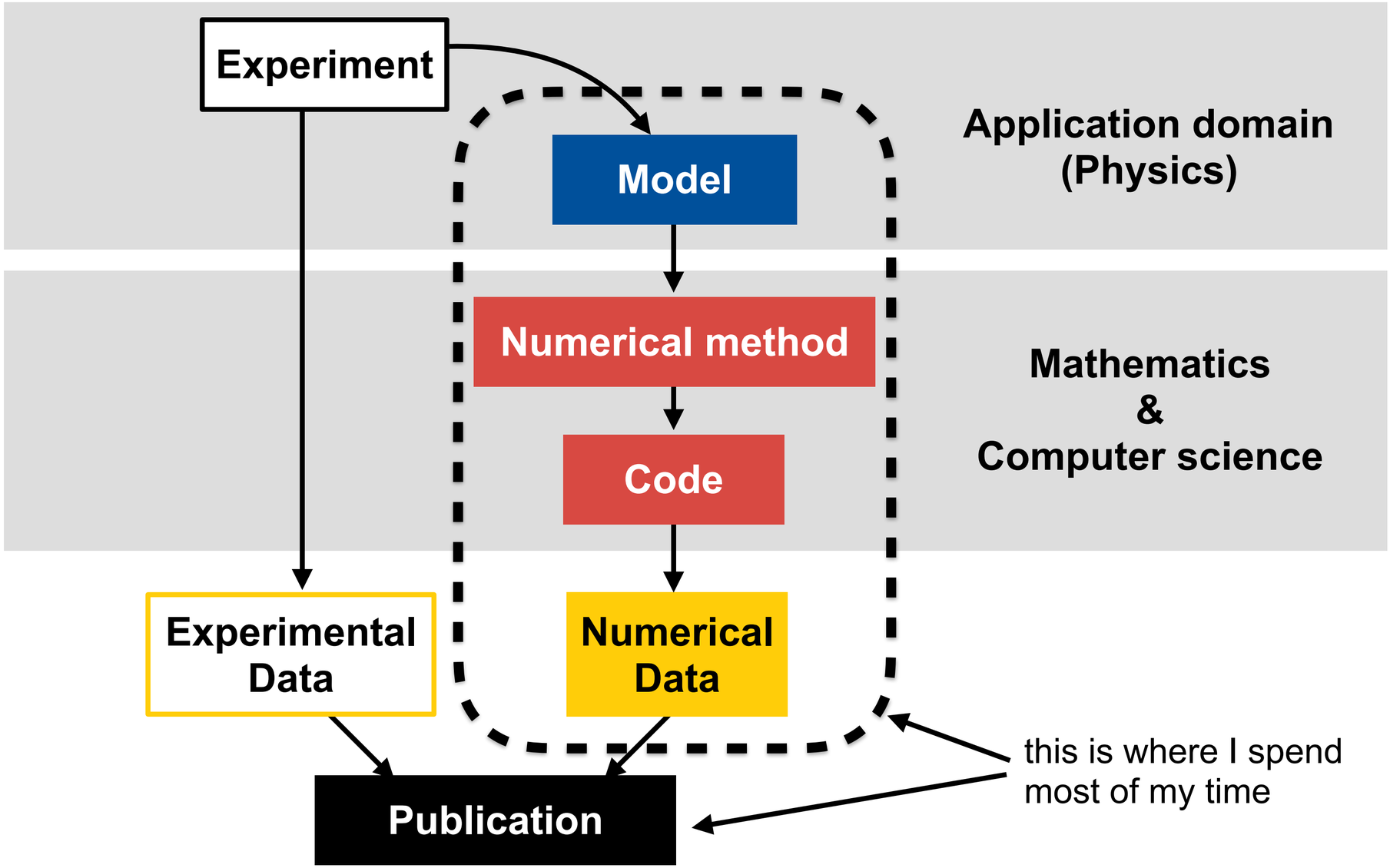
- **Outline**
  - Day-to-day organisational challenges
  - Further data-management activities
  - ➡ 13 best practices + other coping mechanisms
  - ➡ Illustrated by means of a project completed in 2022

    [OM, A. Demircan; *SoftwareX* 20 (2022) 101232]

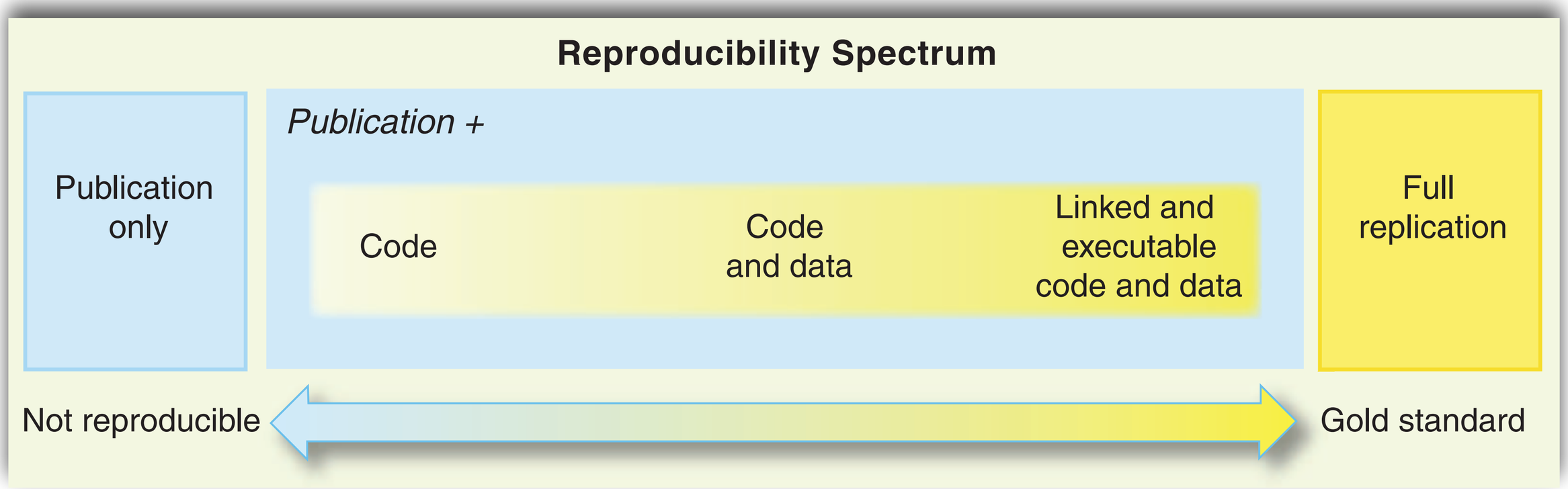

Contents lists available at ScienceDirect
SoftwareX
journal homepage: www.elsevier.com/locate/softx
ELSEVIER
Original software publication
GNLStools.py: A generalized nonlinear Schrödinger Python module implementing different models of input pulse quantum noise

Oliver Melchert*, Ayhan Demircan
Leibniz Universität Hannover, Institute of Quantum Optics (IQO), 30167 Hannover, Germany
Cluster of Excellence PhoenixD (Photonics, Optics, and Engineering – Innovation Across Disciplines), Hannover, Germany

Where do I get my data - typical project life-cycle



What motivates data-management for me?



[D. Peng; *Reproducible Research in Computational Science*; Science 334 (2011) 1226]

- **Reproducibility** is key!
  - Standard by which scientific claims are judged
  - Helps others to build upon my work
  - ➡ Heightens the ***impact*** of my work

- **Research data-management (RDM)**
  - The actions I take to make it *easier* for others to *reproduce* my work
  - ➡ Good scientific practice

User query on GitHub provided opportunity for concise project

Adding noise and calculating the coherence #3

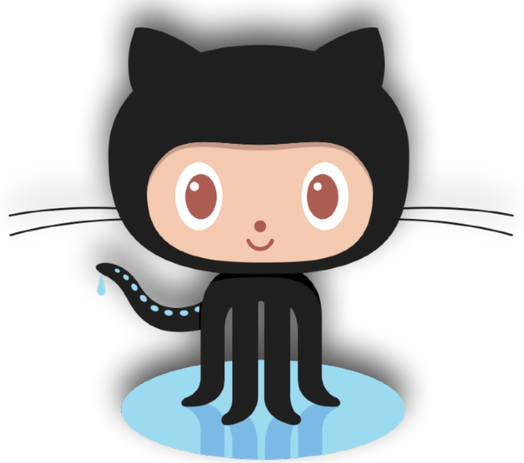
Closed [redacted] opened this issue on Nov 17, 2021 · 4 comments

[redacted] commented on Nov 17, 2021 ···

First I would to thank the authors for their remarkable effort and for making it publicly accessible
My question is the following:
Considering simulating supercontinuum generation in optical fibers. How to add noise to the initial pulse and calculate the first order coherence. It will very nice if you add a small example of that.
Thank you again

omelchert commented on Nov 22, 2021 Owner ···

Thank you for this suggestion. A small extension module allowing to add noise to the initial condition and calculate coherence properties is already underway! I will keep you posted on the progress.



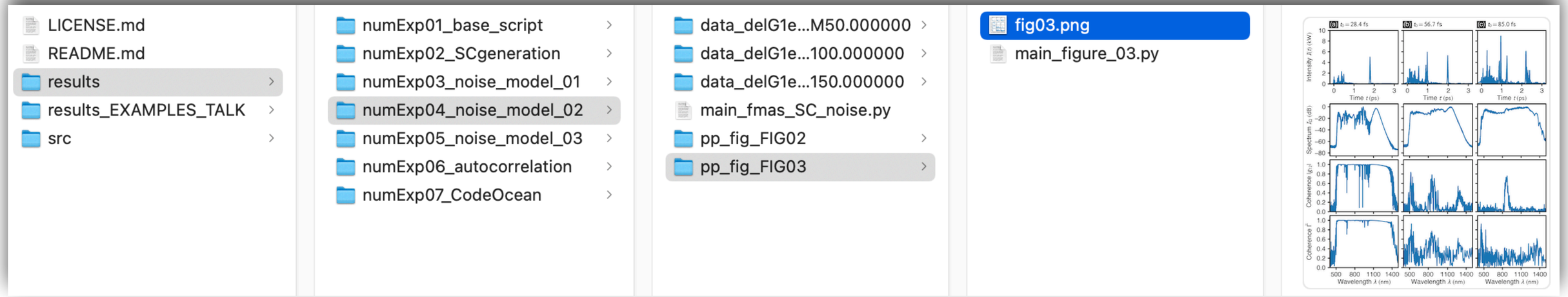
https://github.com

- Requirements
  - Modelling
  - Software engineering
  - Data analysis
  - ➡ Code & data project

Day-to-day organisational challenges

Poor organisational choices can result in slow progress

- Similar folder layout for all projects
  - Data is easily searchable / findable
  - Helps others to navigate your project
  - ➡ Agree on standard of how to do this
- **Best practice 01** - Project `root` folder:
  - Chronological order within "results" directories
  - ➡ Reveals chronological order of project
- **Best practice 02** - Readme file:
  - Overview of project
  - Amended throughout project life-cycle
  - ➡ Valuable when you collaborate with others
  - ➡ Basis for data management plan (DMP)
- **Best practice 03** - Project subfolders:
  - Logical order within project subfolders



[W. Noble; *A Quick Guide to Organizing Computational Biology Projects*; PLoS Comp. Biol. 5 (2009) e1000424]

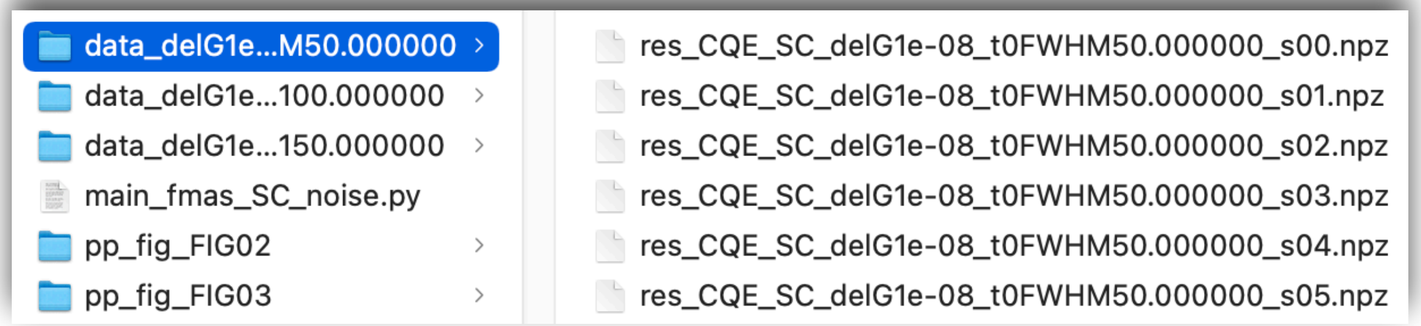
Day-to-day organisational challenges

- **Best practice 04** - Keep it “tidy“:
  - Well documented
  - Modular
  - Easy to read and use
  - ➡ Agree on standards of how to do this

```
    Returns: (du)
        du (1D numpy-array, cplx floats): instance of time-domain noise
    """
    np.random.seed(s0)
    N01 = np.random.normal
    dt = t[1]-t[0]

    # -- REPRESENTATIVE ENERGY OF PHOTON IN BIN
    e0 = hBar*w0                                # (J)
```

[B. Lee; *Ten simple rules for documenting scientific software*; PLoS Comp. Biol. 14 (2018) e1006561]

- **Best practice 05** - Filenames:
  - Choose meaningful filenames
  - ➡ Prevents accidental overwriting of files
  - ➡ Reveals what data is contained in file



- **Best practice 06** - Data format:
  - *Private* data: Choose any data-format you fancy!
  - *Public* data: Choose format that can be read by many programming languages
  - ➡ Good general choice for structured data: HDF5 **https://www.hdfgroup.org**

Day-to-day organisational challenges

- **Best practice 07** - Metadata:
  - Embed metadata *within* your files
  - Include:
    - Information about software environment
    - "Natural language" description of data
  - Ensure it can be read by humans and machines!

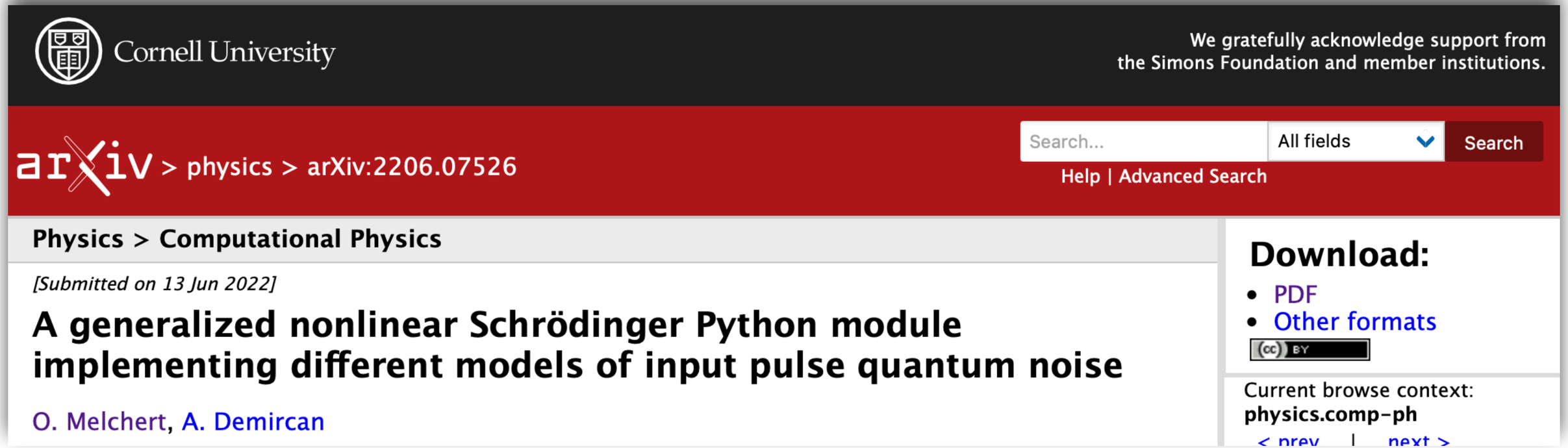
➡ Establishes data provenance

```python
import sys, numpy

path = sys.argv[1]
info = numpy.load(path)['info']

print(info)
```

```
[00 -- I:INFO, D:DATA
 I01 OS-USER: /Users/melchert
 I02 OS-ENV: ('Darwin', 'Olivers-iMac', '18.7.0', 'Darwin Kernel Version 18.7.0: Tue Aug 20 16:57:14 PDT 2
 I03 OS-PID: 28769
 I04 FILE: main_NLPM750_propagationDynamics_qNoise.py
 I05 VERSION: 1.1
 I06 DATE: 2020-05-13 18:38:40.627047
 I07 FNAME: GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSkip50_w02.092929_t0200.000
 D01 z (numpy-array, ndim=1): z-axis, i.e. propagation direction axis
 D02 t (numpy-array, ndim=1): time axis
 D03 w (numpy-array, ndim=1): anglular frequency axis
 D04 Aw (numpy-array, ndim=2): frequency components of field
```

How to disseminate results prior to peer review?

- **Best practice 08** - Post a preprint
  - Establishes priority
  - Broadcasts results early-on
  - Allows for community feedback
  - ➡ Permanent part of scientific record



https://arxiv.org/abs/2206.07526

How to make code & data citable?

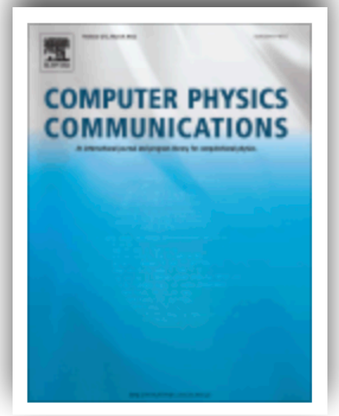
- **Scientific content citation** problem

  There **used to be** no standards for content other than articles!

- Journals that support the **FAIR** principle:
  - **F**indable, **A**ccessible, **I**nteroperable, **R**eusable
  - ➡ Provides literature references for research artefacts

(Open Access)

(Open Access)



Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

GNLStools.py: A generalized nonlinear Schrödinger Python module implementing different models of input pulse quantum noise

Oliver Melchert *, Ayhan Demircan

Leibniz Universität Hannover, Institute of Quantum Optics (IQO), 30167 Hannover, Germany
Cluster of Excellence PhoenixD (Photonics, Optics, and Engineering – Innovation Across Disciplines), Hannover, Germany

ARTICLE INFO

Article history:
Received 14 June 2022
Received in revised form 15 September 2022
Accepted 11 October 2022

Keywords:
Generalized nonlinear Schrödinger equation
Quantum noise
Spectral coherence
Python

ABSTRACT

We provide Python tools enabling numerical simulation and analysis of the propagation dynamics of ultrashort laser pulses in nonlinear waveguides. The modeling approach is based on the widely used generalized nonlinear Schrödinger equation for the pulse envelope. The presented software implements the effects of linear dispersion, pulse self-steepening, and the Raman effect. The focus lies on the implementation of input pulse shot noise, i.e. classical background fields that mimic quantum noise, which are often not thoroughly presented in the scientific literature. We discuss and implement commonly adopted quantum noise models based on pure spectral phase noise, as well as Gaussian noise. Coherence properties of the resulting spectra can be calculated. We demonstrate the functionality of the software by reproducing results for a supercontinuum generation process in a photonic crystal fiber, documented in the scientific literature. The presented Python tools are open-source and released under the MIT license in a publicly available software repository.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00165
Legal Code License	MIT License
Code versioning system used	none
Software code languages, tools, and services used	Python, GitHub
Compilation requirements, operating environments & dependencies	The provided software requires Python, numpy and scipy. The provided examples need Pythons `matplotlib` for figure generation.
If available Link to developer documentation/manual	Documentation provided within code
Support email for questions	melchert@iqo.uni-hannover.de

1. Introduction

The propagation of laser pulses in nonlinear waveguides supports the generation of supercontinuum spectra [1–3]. Starting from a spectrally narrow input pulse, the interplay of linear and nonlinear effects induces tremendous spectral broadening, yielding flat spectra that can extend from the violet to the infrared [4]. Such effects can be achieved, e.g., in photonic crystal fibers (PCFs) [5,6], wherein supercontinuum spectra can be produced using ∼100 fs-duration pulses, peak powers ∼10 kW and propagation lengths on the order of 1 m [4]. The resulting broad, flat spectra with high spectral density find application, e.g., in optical frequency metrology [7], and optical technologies [2].

A flexible theoretical framework for studying the complex physical processes associated with the generation of supercontinuum spectra is provided by the generalized nonlinear Schrödinger equation (GNLS) [1]. In order to model the propagation dynamics of laser pulses it combines the effects of linear dispersion, pulse self-steepening [8,9], and the Raman effect [10]. This accounts for various processes that support the generation of widely

* Corresponding author at: Leibniz Universität Hannover, Institute of Quantum Optics (IQO), 30167 Hannover, Germany.
E-mail addresses: melchert@iqo.uni-hannover.de (Oliver Melchert), demircan@iqo.uni-hannover.de (Ayhan Demircan).

https://doi.org/10.1016/j.softx.2022.101232

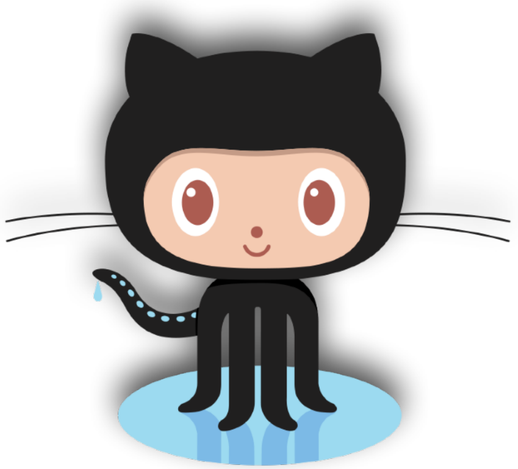
[OM, B. Roth, U. Morgner, A. Demircan; *SoftwareX* 10 (2019) 100275]
[OM, A. Demircan; *SoftwareX* 15 (2021) 100741]
[OM, A. Demircan; *Comp. Phys. Commun.* 273 (2022) 108257]
[OM, A. Demircan; *SoftwareX* 20 (2022) 101232]

How to make code accessible?

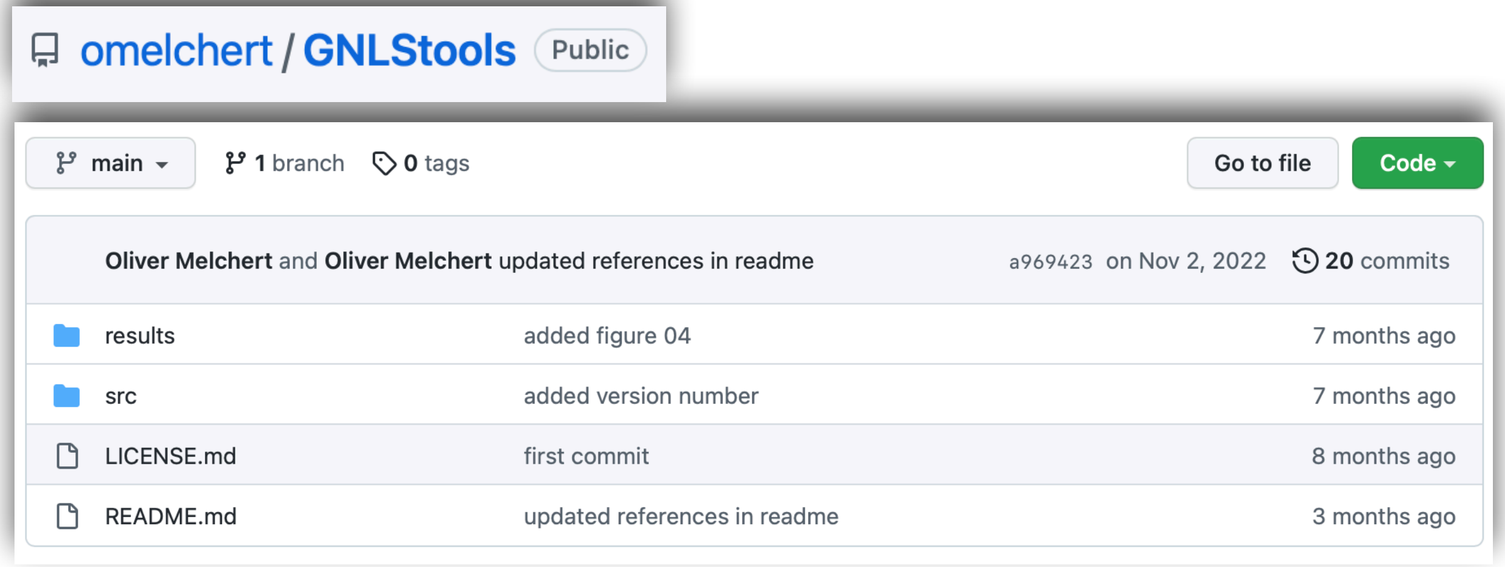
- **Code availability** problem

  There are no standards that clarify how to make code available for others!

- **Best practice 09** - Git[Hub/Lab]:
  - Version control platform allowing to develop/share code
  - ➡ Helps making code externally available (repositories can also be kept private!)
  - ➡ File size limitations: 100 Mb/file (Git), 2 GB/file (Git-LFS; Large File Service)



https://github.com

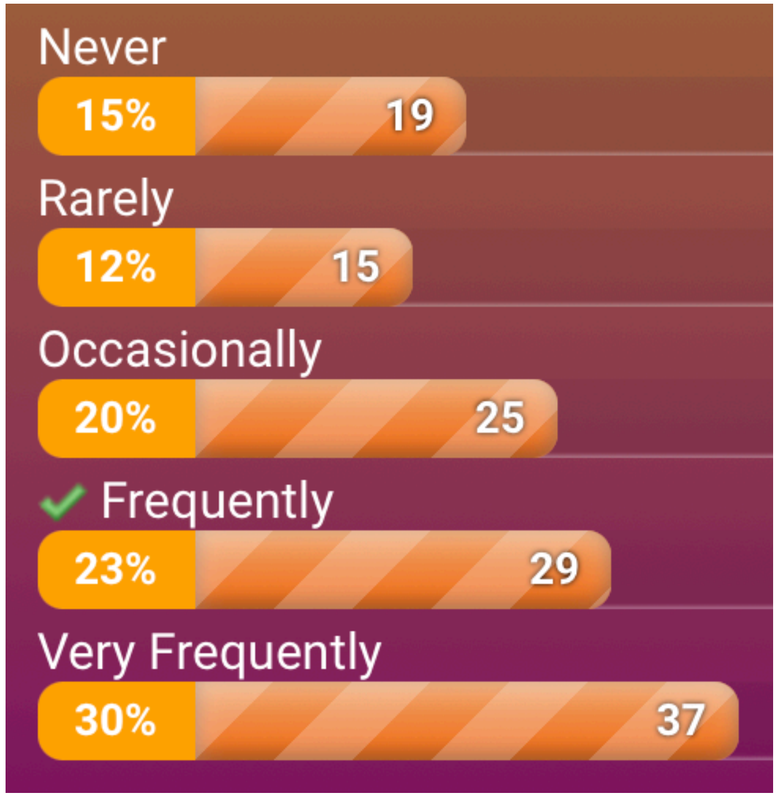


[Y. Perez-Riverol *et al.*; *Ten Simple Rules for Taking Advantage of Git and GitHub*; PLoS Comp. Biol. 12 (2016) e1004948]

Do readers really want access to code?

- PLOS Computational Biology survey (conducted in 2021)
  https://plos.org/open-science/open-code/

- Do you consult public code?



- Aims when consulting public code

70% Aiding understanding of the article	63% Verifying or exploring code directly	48% Reusing or repurposing code
21% Better understanding methods	21% Assessing the quality of the research	12% Replicating the study using their own data

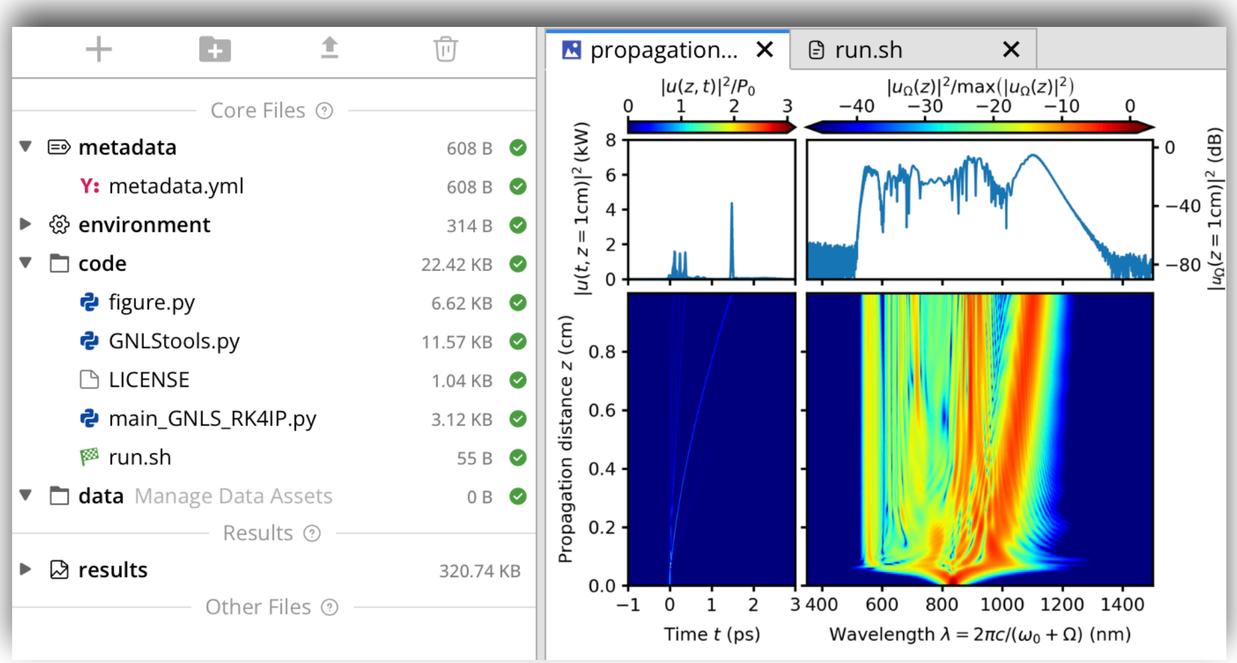
How to make code easily interoperable?

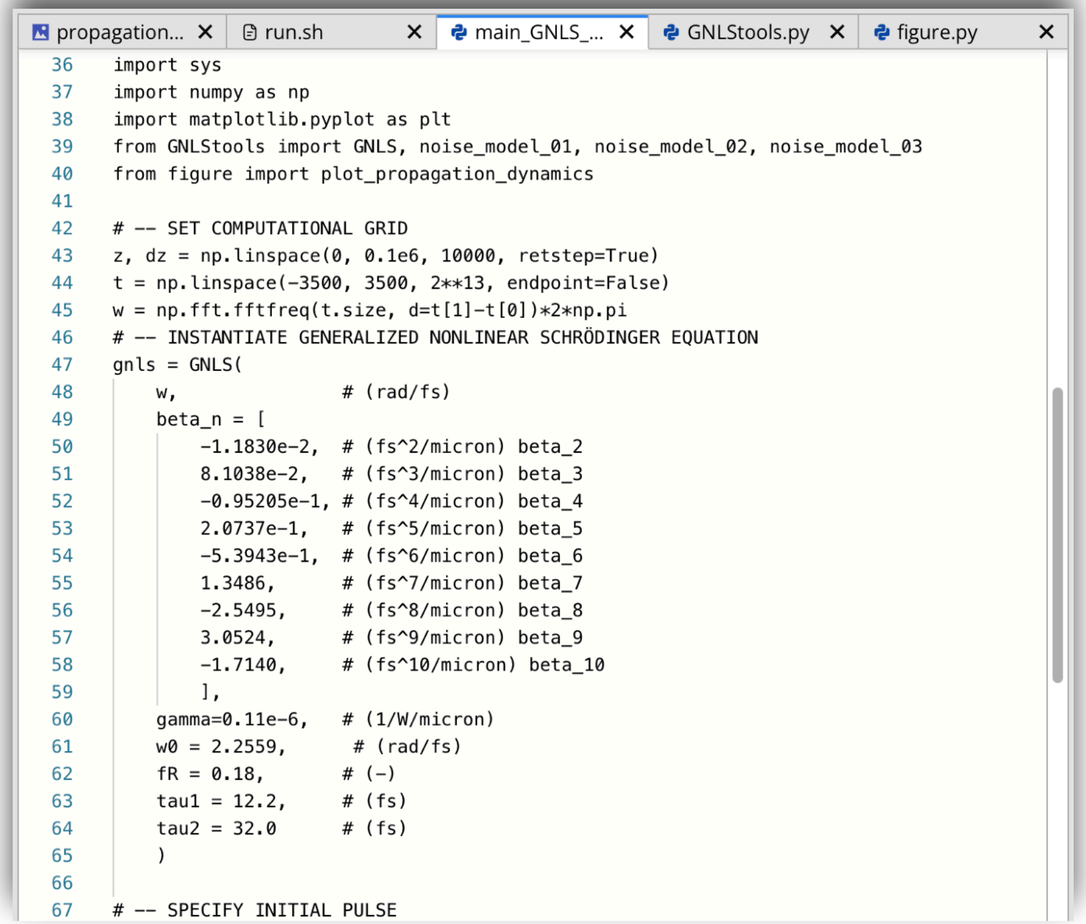
- **Interoperability** problem

- **Best practice 10** - Share on custom environment:
  - SoftwareX partners with Code Ocean
  - ➡ Enables collaborative computational research
  - ➡ Lets a user test your software without installing it

CODE OCEAN

https://codeocean.com

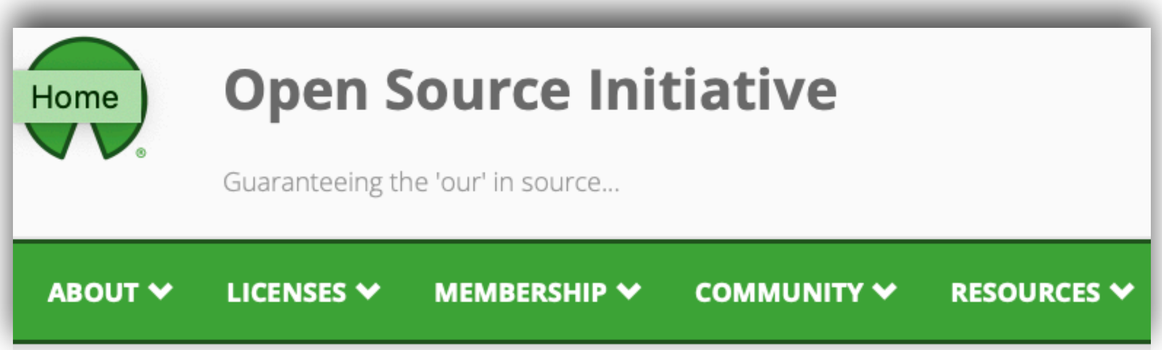




https://codeocean.com/capsule/4658074

Set terms on which software might be used!

- **Best practice 11** - Provide a license:
  - Allows others to reuse your code
  - Clarify who owns the *intellectual property* (IP) rights
  - ➡ Facilitates access to software (as well as restricts it)



https://opensource.org

- Example of how I did this for the discussed project
  - IP owned by PhoenixD and me
  - I use the MIT-license
  - Licence header:

```
Copyright   2022 PhoenixD
            Contributor: Oliver Melchert
            Affiliation: Theoretical Optics and Computational Photonics Group,
                         Institute of Quantum Optics,
                         Leibniz Universität Hannover
```

[A. Morin *et al.*; *A Quick Guide to Software Licensing for the Scientist-Programmer*; PLoS Comp. Biol. 8 (2012) e1002598]

How to make auxiliary data/media citable?

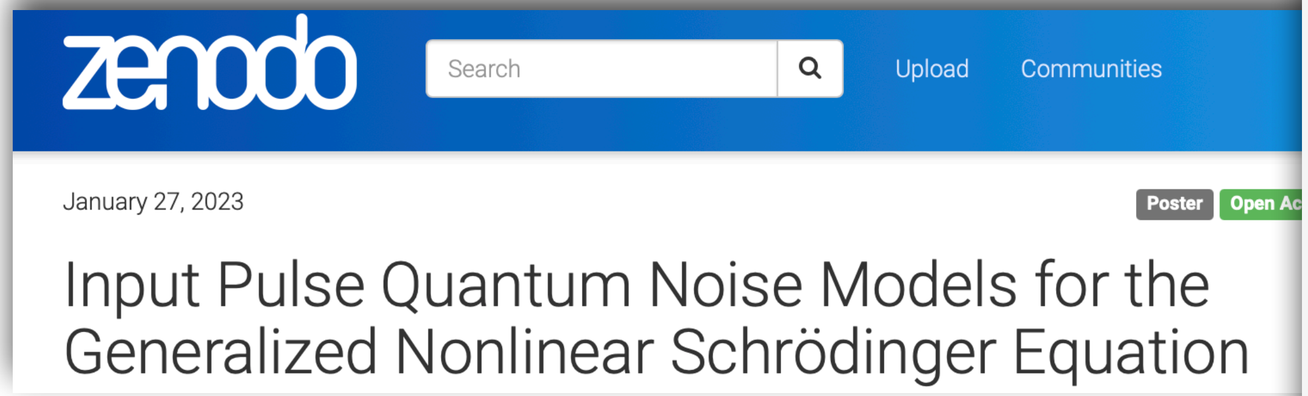
- **Auxiliary data/media availability** problem

  There are no standards that clarify how to make data available!

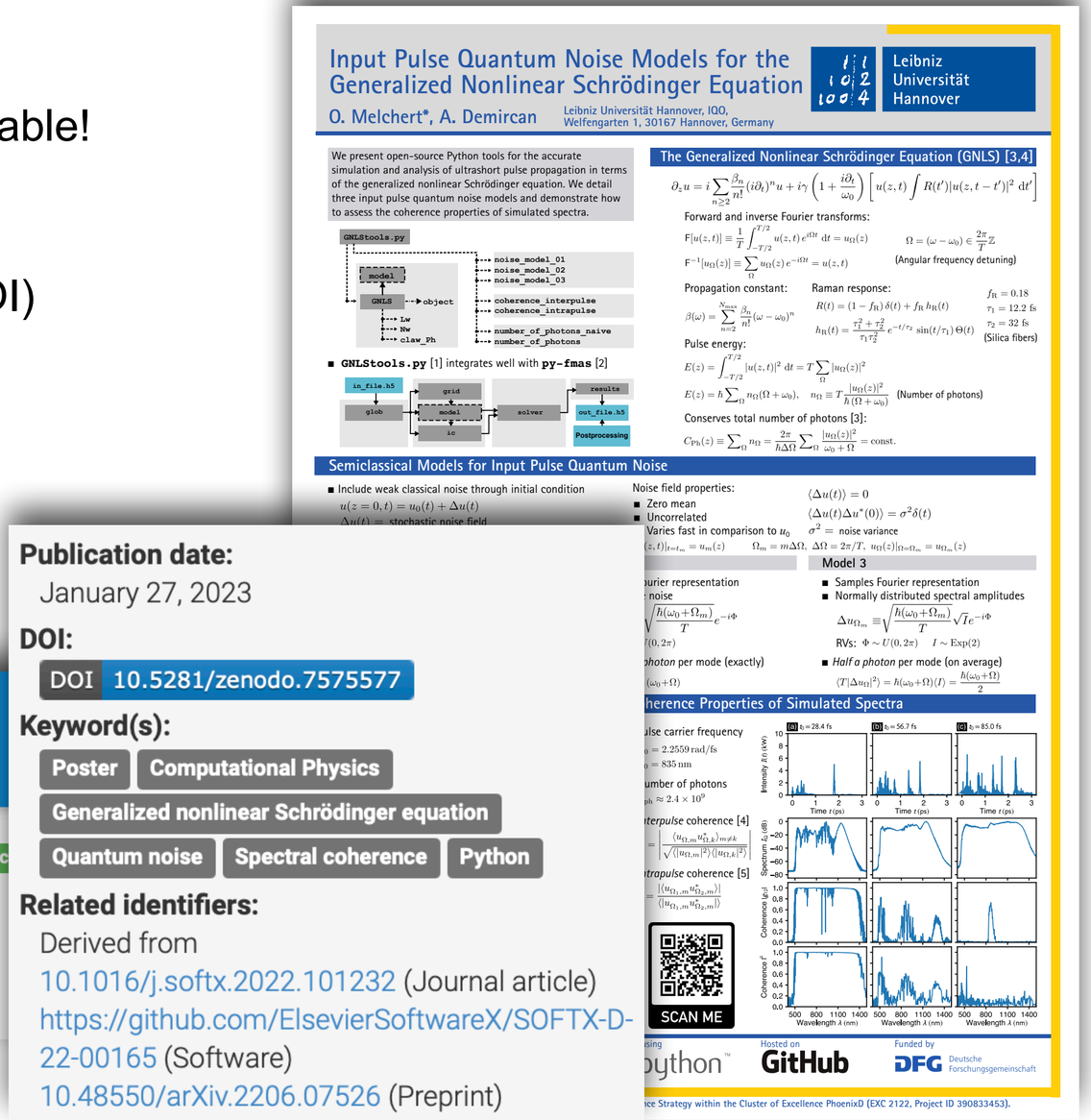
- **Best practice 12** - DOIs:
  - Establish provenance by using Digital Object Identifier (DOI) provided by data hosting platforms like Zenodo or figshare
    https://zenodo.org **https://figshare.com**

  ➡ Allows to share code & data and make it citable via DOIs

  ➡ Provides storage and preservation strategy for your data



(max. 50 GB per data set)



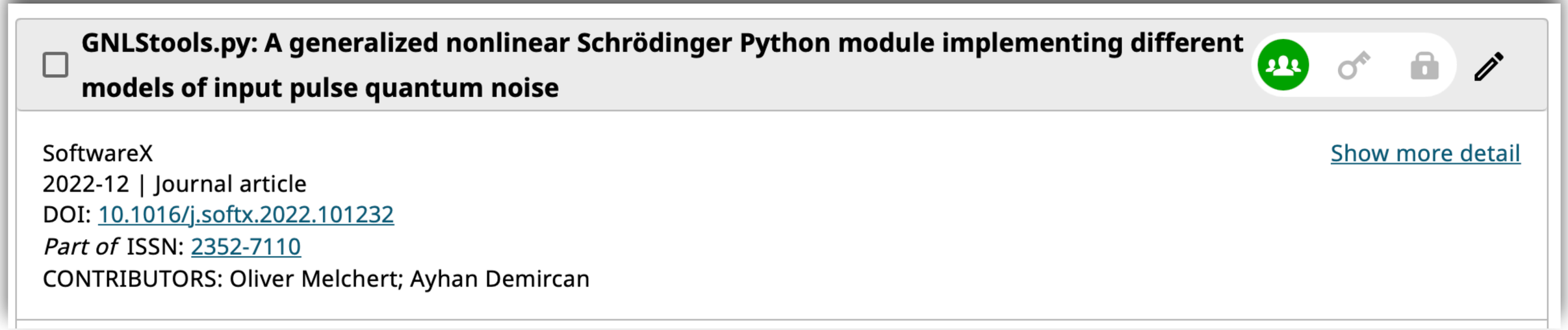
How to claim ownership of your publications?

- **Accurate attribution of scholarly research output** problem

- **Best practice 13** - ORCID:
  - Open Researcher and Contributor ID
  - ➡ Provides unique, persistent identifier to researchers
  - ➡ Creates permanent record or research (not limited to publications)

ORCID
Connecting research and researchers

https://orcid.org



Do your very best — and talk [tweet] about it!

https://twitter.com

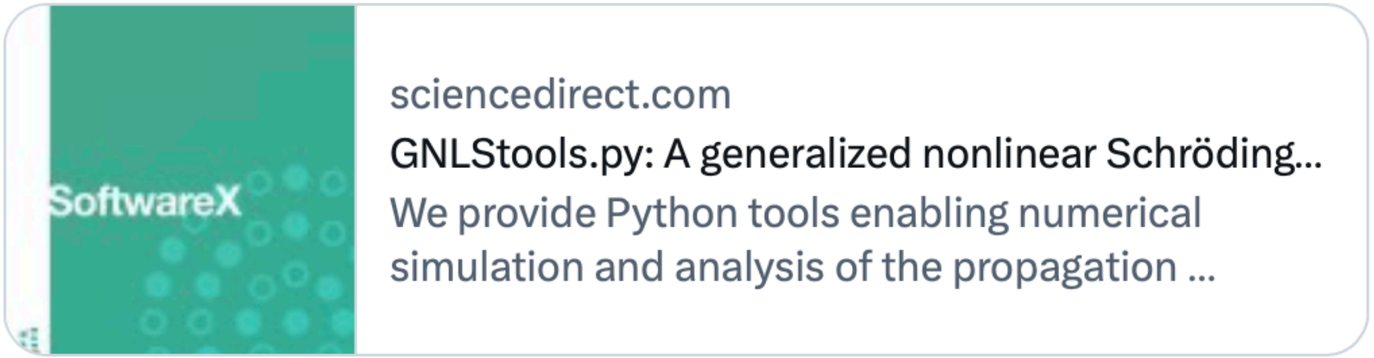
Oliver Melchert @OliverMelchert · Nov 2, 2022

Especially happy about this one, originating from a user-query on gitHub!

GNLStools.py: A generalized nonlinear Schrödinger Python module implementing different models of input pulse quantum noise

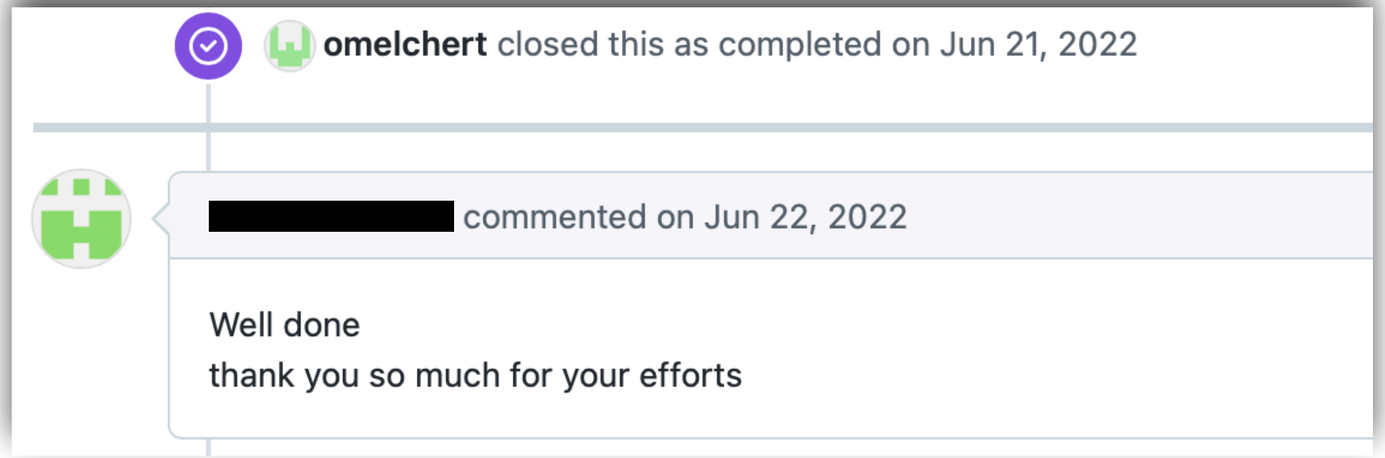
doi.org/10.1016/j.soft...

@SoftXJournal @github #Python #Physics



Summary

- RDM strategies helping to reproduce your work with ease
  - Best practices related to
    - Day-to-day organisational challenges
    - Further data-management activities
    - Quality control (not discussed)
  - Agree on "how to do things" (when there are no RDM guidelines)
  - Implementing these strategies takes time and effort
  - ➡ Develop a culture that values RDM



DFG Deutsche Forschungsgemeinschaft

(EXC 2122, projectID 390833453)

Backup 01 — Questions asked in previous talks

- **Is Github a viable way to store data and what about self-hosted Git solutions (Gitlab etc.)?**
  - In principle yes, but it depends on the type of data
  - I use it to host code, only!
  - GitHub has file-size limits:
    - it blocks pushes that exceed 100 MB
    - Large file storage solution

  - Zenodo (max. 50 GB per data set)
    https://zenodo.org/
  - Consider using seafile (quota depends on project)
    https://seafile.projekt.uni-hannover.de/
  - Large data: consider High-seas (file sizes up to TB)
    https://high-seas.projekt.uni-hannover.de/

Using Git LFS, you can store files up to:

Product	Maximum file size
GitHub Free	2 GB
GitHub Pro	2 GB
GitHub Team	4 GB
GitHub Enterprise Cloud	5 GB

https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage

[Y. Perez-Riverol *et al.*; *Ten Simple Rules for Taking Advantage of Git and GitHub*; PLoS Comp. Biol. 12 (2016) e1004948]

Backup 02 — Questions asked in previous talks

- **Should the source code be made available to everyone and should it be licensed under open source licenses?**
  - Depends on the data management guidelines
  - Why you should do it (in my opinion):
    - Freely provided code, whatever the quality, enables others to engage with your work
    - Making it available allows you to also cite it!
  - If your data management guidelines allow to do so, use an open source license. Code is meant to be used exactly as it is written, so licenses help to avoid plagiarism issues!

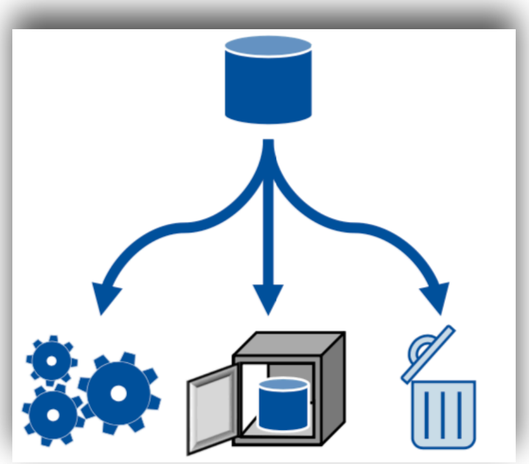
[N. Barnes; *Publish your computer code: it is good enough*; Nature 467 (2010)753]

[A. Morin *et al.*; *A Quick Guide to Software Licensing for the Scientist-Programmer*; PLoS Comp. Biol. 8 (2012) e1002598]

Backup 03 — Questions asked in previous talks

- **In what way should research data (raw data, code, I/O-parameters) be stored?**
  - Store data in a way so that your results can be reproduced with ease!
  - Find compromise between
    - *Cost in time:* time it takes to reproduce data when given the code + parameters
    - *Cost in disk space:* amount of space required to store *all* research data
  - If you can afford to pay some *cost in time*, you can keep the *cost in disk space* pretty small
  - ➡ Applies best to computer simulation studies
  - ➡ In laboratory experiments: *high cost in time* often equals *high cost in person-power;* then you dont want to repeat experiments and keep all the data ... even if the whole experiment went wrong!
  - Bugs are also research data!
  - ➡ Document them very well and keep track of which program versions where prone to it!

Backup 04 — When to **not** store data?



www.fdm.uni-hannover.de/en
(Which data to keep long-term?)

- **Best practice** - Know what data to preserve:
  - You have leverage to select what data to preserve!
  - Many files + single file can be reproduced quickly?
  - Don't preserve the data, but keep a script that details how to reproduce it!

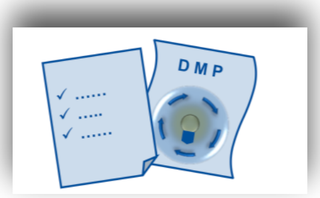
  ➡ Conserves space required to store the data

Example of when I don't store the data:

```
[melchert@Olivers-iMac:[numExp02_Revision_01]: ls data/
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
GNLSE_BlowWood_tMax8000.000000_Nt16384_zMax1000000.000000_Nz10000_nSk
[melchert@Olivers-iMac:[numExp02_Revision_01]: du -h data/
719M    data/
```

+ it takes < 30 minutes to reproduce the data!

Backup 05 — Maintain a data management plan (DMP)



- What is a DMP?

  A DMP comprises all your data management activities ... in written form!

www.fdm.uni-hannover.de/en
(Tools for developing a DMP)

- **Best practice** - Maintain a DMP:
  - Describes how you treat data during the project
  - Describes the roles and responsibilities of collaborators
  - Covers entire project life-cycle
    - Data collection
    - Data organisation
    - Quality control (we skipped this!)
    - Data storage and backup (we also skipped this!)
    - Data documentation
    - Data preservation
    - Sharing with others
  - Might be used to evaluate a projects merit
  - Basis for the DMP is usually the project Readme file!

https://dmponline.dcc.ac.uk

[W. Michener; *Ten Simple Rules for Creating a Good Data Management Plan*; PLoS Comp. Biol. 11 (2015) e1004525]