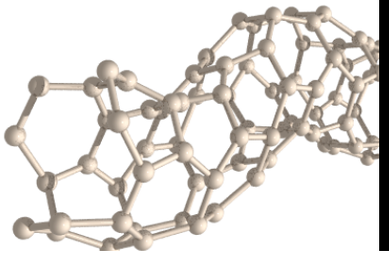
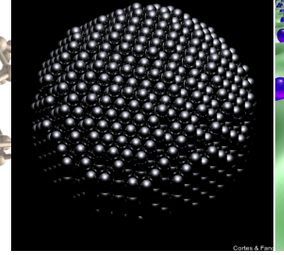
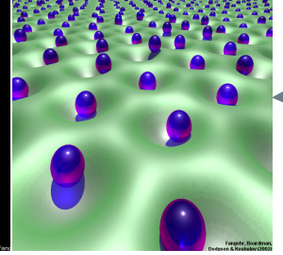
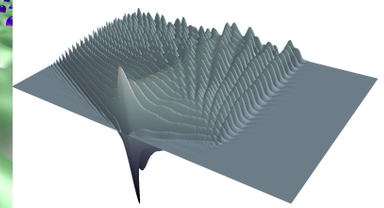
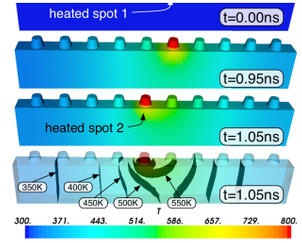
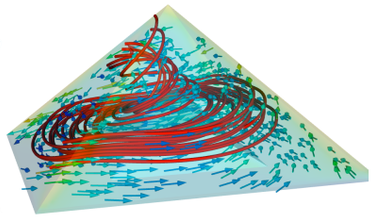
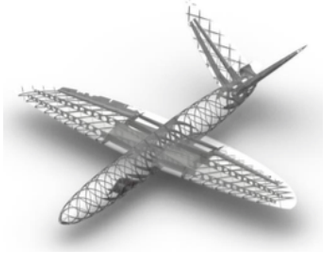
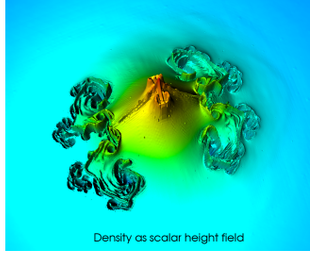






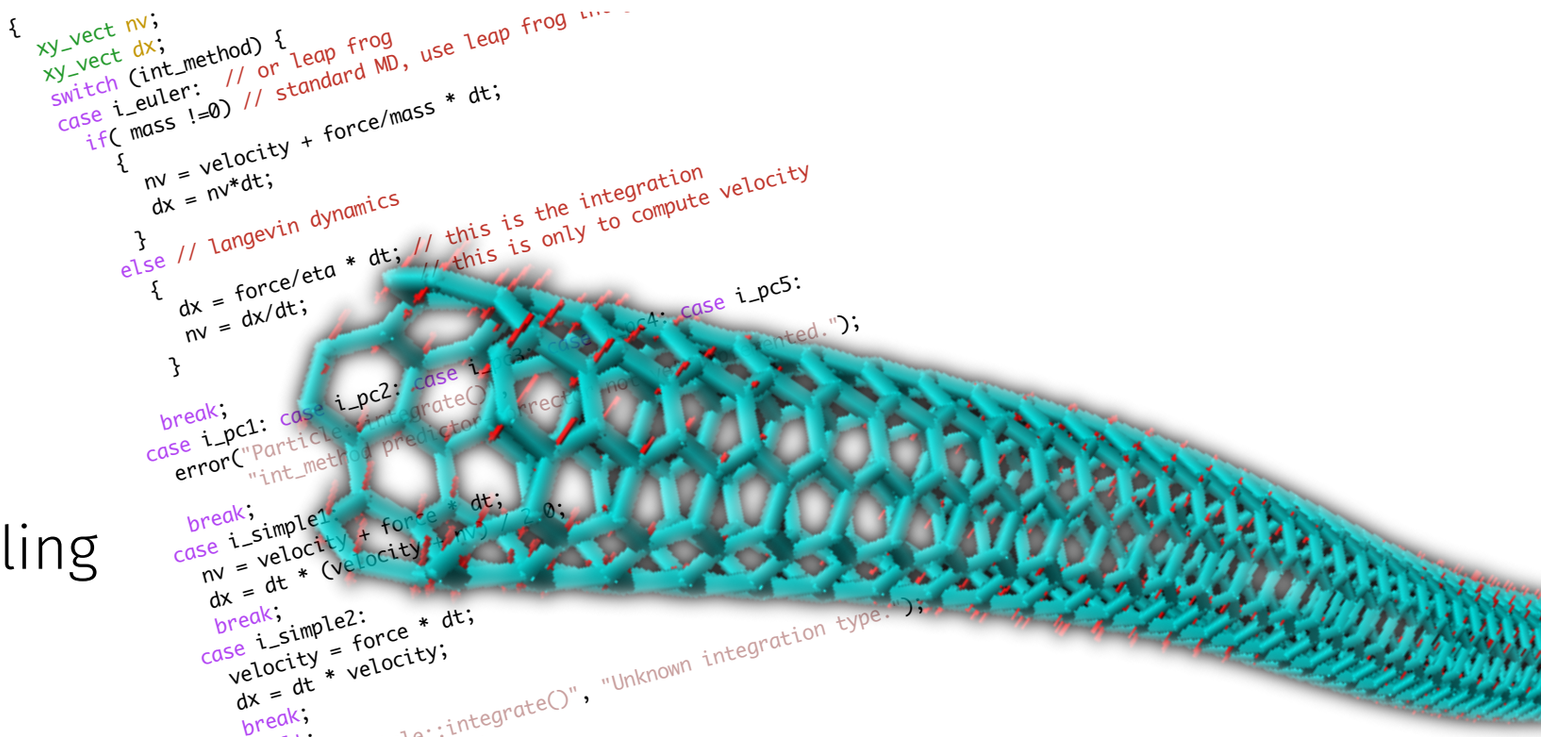




Software for Science

Hans Fangohr
Data Analysis Scientist
European XFEL GmbH
Germany

Professor of Computational Modelling
University of Southampton
United Kingdom

Copenhagen, 16 November 2017



European XFEL

Outline

- Introduction
- Computational Science & Challenges
- Essential tools (Software engineering for Science)
- Useful tools (Jupyter Notebook)
- Political and educational approaches

Hans Fangohr

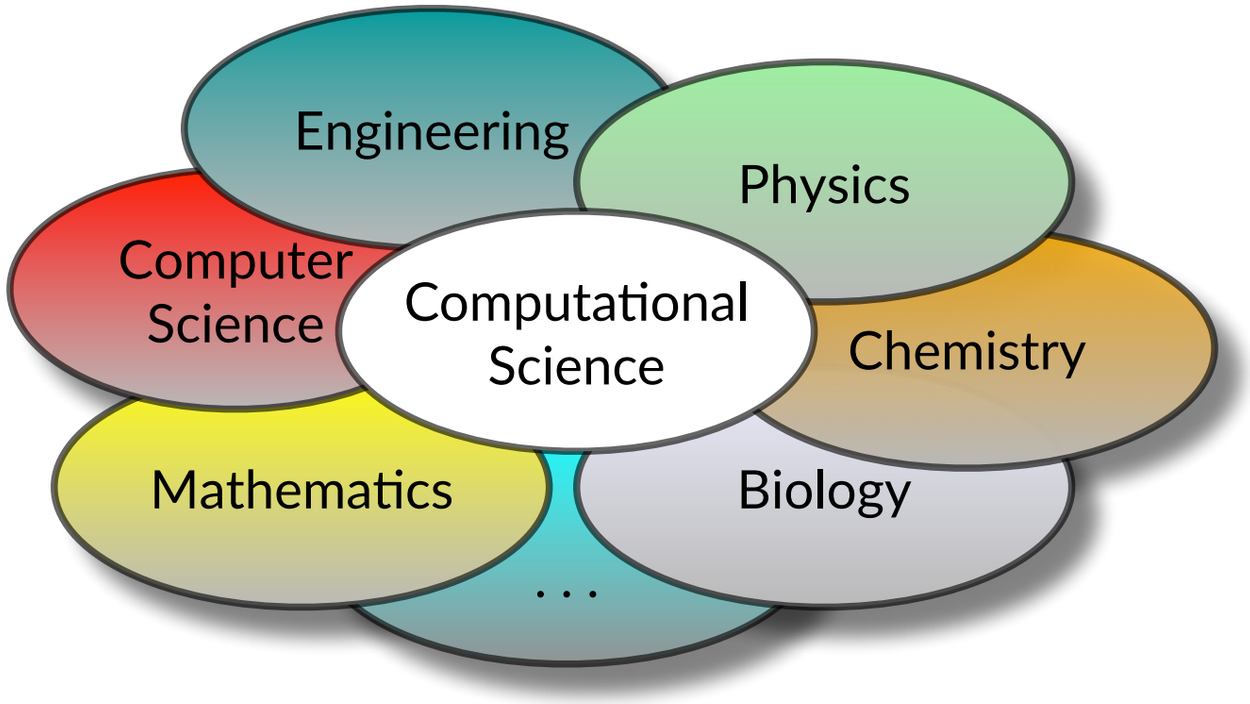
- Undergraduate degree “Diplomphysiker” with computer science and applied mathematics in Hamburg (1994-1999)
- PhD in High Performance Computing group in Computer Science Department in Southampton, United Kingdom (1999-2002)
- Lecturer, Senior Lecturer in Computational Methods (2002-2010)
- Professor of Computational Modelling (since 2010), Southampton, United Kingdom
- From 2017, Senior Data Analysis Scientist at European XFEL GmbH (http://xfel.eu), Germany

Coding day with research group at Southampton

Hans Fangohr
E1.145
http://fangohr.github.io
@ProfCompMod

What is computational science?

- Use of computers to support and enable scientific insight
  - Data processing
  - Data analysis
  - Visualisation
  - Simulation
  - Virtual design
  - Virtual design optimisation



- Interdisciplinary / no training

Computational Science – the third pillar

- Computational science complements theory and experiment
- Scientists typically spend 30% or more of their time developing software [1, 2]
- 90% or more of them are primarily self-taught [1, 2]
- 70% of researchers say: it is impossible to conduct research without research software [3]

[1] Hannay JE, Langtangen HP, MacLeod C, Pfahl D, Singer J, et al.. (2009) How do scientists develop and use scientific software? In: Proceedings Second International Workshop on Software Engineering for Computational Science and Engineering. pp. 1–8. doi:10.1109/SECSE.2009.5069155.

[2] Prabhu P, Jablin TB, Raman A, Zhang Y, Huang J, et al.. (2011) A survey of the practice of computational science. In: Proceedings 24th ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis. pp. 19:1–19:12. doi: 10.1145/2063348.2063374.

[3] December 2014: http://tinyurl.com/ooajs7m

Computational Science also relevant for industry

- Industry: project partners contributing €4m to €12m Centre for Doctoral Training in Computational Modelling
- Founder and director (2013-2017): Hans Fangohr
- http://ngcm.soton.ac.uk

Computational Science “gone wrong” examples

Reinhart, Rogoff, and the Excel Error That Changed History

- Significant error in 2010 research paper "Growth in a Time of Debt" [1] which has been widely cited to justify budget-cutting [2]
- The Harvard professors had accidentally only included 15 of the 20 countries under analysis in their key calculation (of average GDP growth in countries with high public debt)." [3]

[0] https://scholar.harvard.edu/files/rogoff/files/growth_in_time_debt_aer.pdf
[1] https://www.bloomberg.com/news/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history
[2] http://www.bbc.co.uk/news/magazine-22223190

RESEARCH ARTICLE

Structure of MsbA from *E. coli*: A Homolog of the Multidrug Resistance ATP Binding Cassette (ABC) Transporters

Geoffrey Chang* and Christopher B. Roth

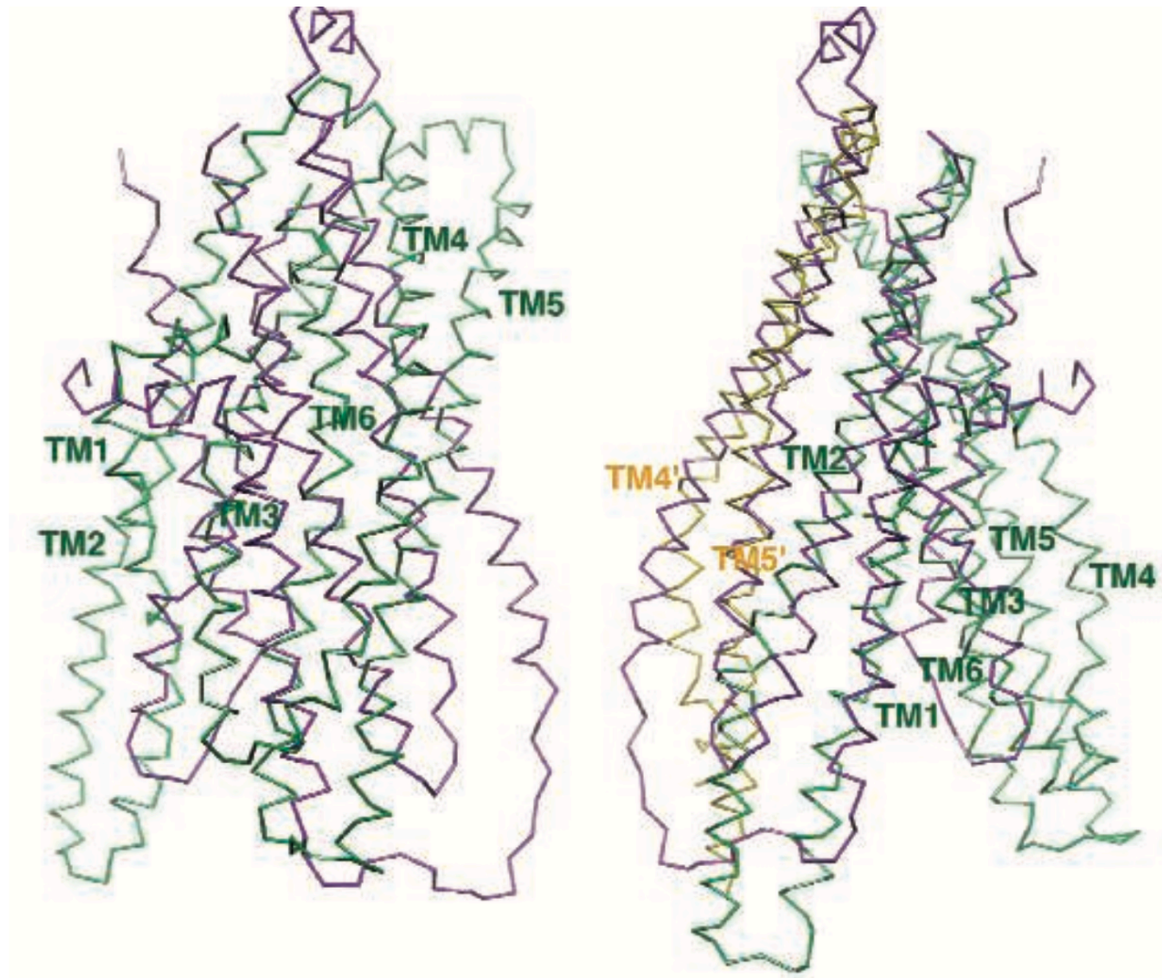
Multidrug resistance (MDR) is a serious medical problem and presents a major challenge to the treatment of disease and the development of novel therapeutics. ABC transporters that are associated with multidrug resistance (MDR-ABC transporters) translocate hydrophobic drugs and lipids from the inner to the outer leaflet of the cell membrane. To better elucidate the structural basis for the "flip-flop" mechanism of substrate movement across the lipid bilayer, we have determined the structure of the lipid flippase MsbA from *Escherichia coli* by x-ray crystallography to a resolution of 4.5 angstroms. MsbA is organized as a homodimer with each subunit containing six transmembrane α-helices and a nucleotide-binding domain. The asymmetric distribution of charged residues lining a central chamber suggests a general mechanism for the translocation of substrate by MsbA and other MDR-ABC transporters. The structure of MsbA can serve as a model for the MDR-ABC transporters that confer multidrug resistance to cancer cells and infectious microorganisms.

coproteins, which have these components fused into a single polypeptide, the msbA gene encodes a half transporter that contains a single membrane spanning region fused with a NBD. MsbA is assembled as a homodimer with a total molecular mass of 129.2 kD. Hydropathy analysis indicates six membrane spanning regions with the NBD located on the cytoplasmic side of the cell membrane (*17*). The primary role of the transmembrane domain is to recognize and transport substrates across the lipid bilayer. The ABC, which is the hallmark of the MDR-ABC transporter family and is located in the NBD, couples the energy of ATP hydrolysis to substrate translocation. Although the NBD structures of the histidine transporter (HisP), the maltose transporter (MalK), the DNA repair enzyme (Rad50), and the branched-chain amino acid transporter from *Methanococcus jannaschii* (MJ1267) have been determined, the structural basis for substrate translocation through the cell membrane is not clear (*18–21*).

The structure of MsbA establishes the general architecture of the MDR-ABC transporter family, and facilitates our understanding of the fundamental flipping mechanism that moves hydrophobic sub-

for Scientists and Engineers, the country's highest honor for young researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

Then the dream turned into a nightmare. In September, Swiss researchers published a paper in *Nature* that cast serious doubt on a protein structure Chang's group had described in a 2001 *Science* paper. When he investigated, Chang was horrified to discover that a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the final protein structure. Unfortunately, his group had used the program to analyze data for other proteins. As a result, on page 1875, Chang and his colleagues retract three *Science* papers and report that two papers in other journals also contain erroneous structures.



Flipping fiasco. The structures of MsbA (purple) and Sav1866 (green) overlap little (*left*) until MsbA is inverted (*right*).

"I've been devastated," Chang says. "I hope people will understand that it was a mistake, and I'm very sorry for it." Other researchers don't doubt that the error was unintentional, and although some say it has cost them time

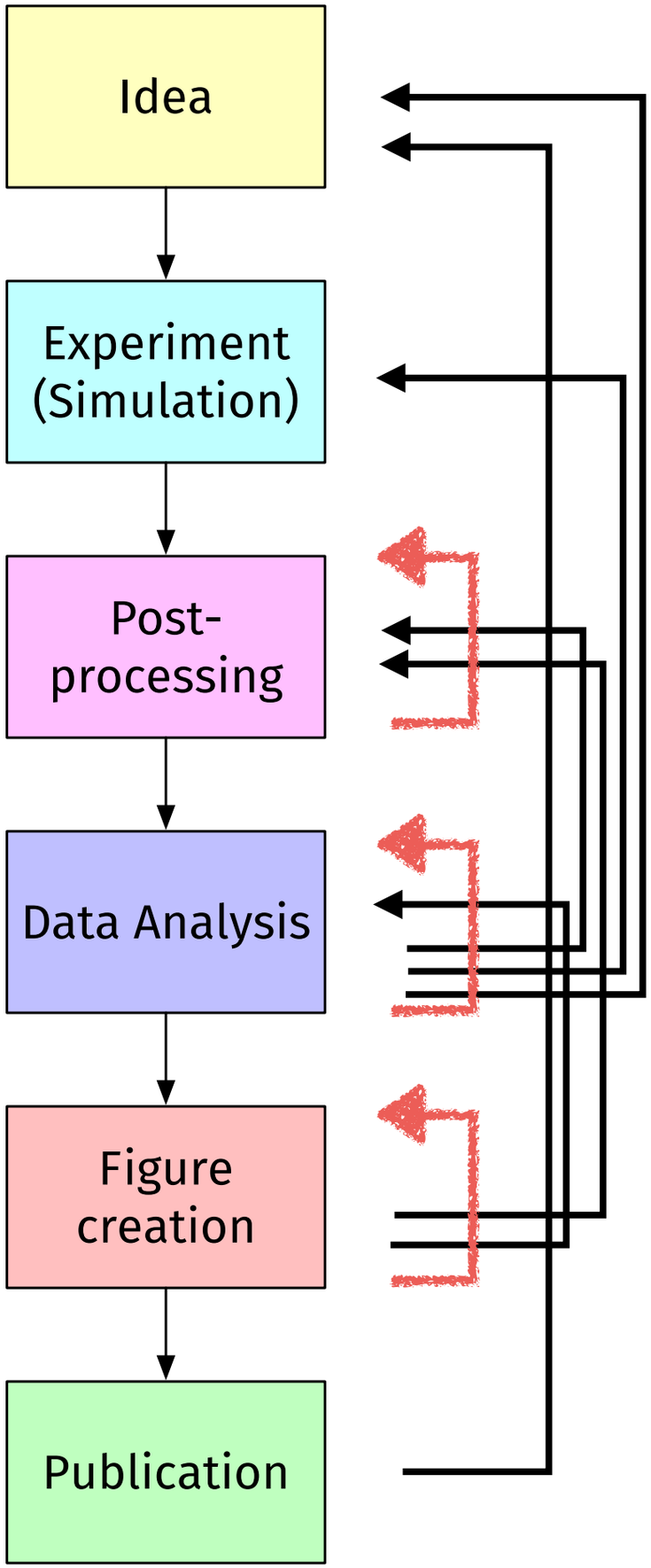
essential biological duties and are of great clinical interest because of their roles in drug resistance. Some pump antibiotics out of bacterial cells, for example; others clear chemotherapy drugs from cancer cells. Chang's MsbA structure was the first molecular portrait of an entire ABC transporter, and many researchers saw it as a major contribution toward figuring out how these crucial proteins do their jobs. That paper

no one else had been able to do." Chang's data are good, Rees says, but the faulty software threw everything off.

Ironically, another former postdoc in Rees's lab, Kaspar Locher, exposed the mistake. In the 14 September issue of *Nature*, Locher, now at the Swiss Federal Institute of Technology in Zurich, described the structure of an ABC transporter called Sav1866 from *Staphylococcus aureus*. The structure was dramatically—and unexpectedly—different from that of MsbA. After pulling up Sav1866 and Chang's MsbA from *S. typhimurium* on a computer screen, Locher says he realized in minutes that the MsbA structure was inverted. Interpreting the "hand" of a molecule is always a challenge for crystallographers, Locher notes, and many mistakes can lead to an incorrect mirror-image structure. Getting the wrong hand is "in the category of monumental blunders," Locher says.

On reading the *Nature* paper, Chang quickly traced the mix-up back to the analysis program, which he says he inherited from another lab. Locher suspects that Chang would have caught the mistake if he'd taken

R, *NATURE* **443**, 180 (2006)



Reproducibility / Scientific Workflow

- Iterative exploration of a problem/data set via computation and intermediate results
- At the end of the study, results must be communicated through a (linear) *narrative* (paper/report/thesis,...).
- Should be reproducible.

Reproducibility in data analysis

- Open package X, use mouse to select functions in menu, then click, drag, and click, and in the end save numbers.txt. Load into Origin or Excel. Click, drag, generate graph, right click, save as ‘`good-numbers.png`’
- In paper: “*We analysed our data using package X to produce figure 1.*”
- Is this reproducible? Can we do exactly the same thing a year later? A day later? Ever?

Challenges for Computational Science

Challenges for software in science

- reproducibility
- testing can be difficult
- ongoing changes throughout the life time of the project
- no incentive to document code or write it well – only publications count
- coders may not have programming and software engineering training
- main code author often leaving after n years (n is small)
- code lost or not publicly available
- fast execution competes with readable and maintainable code
- hardware changes require re-write of codes

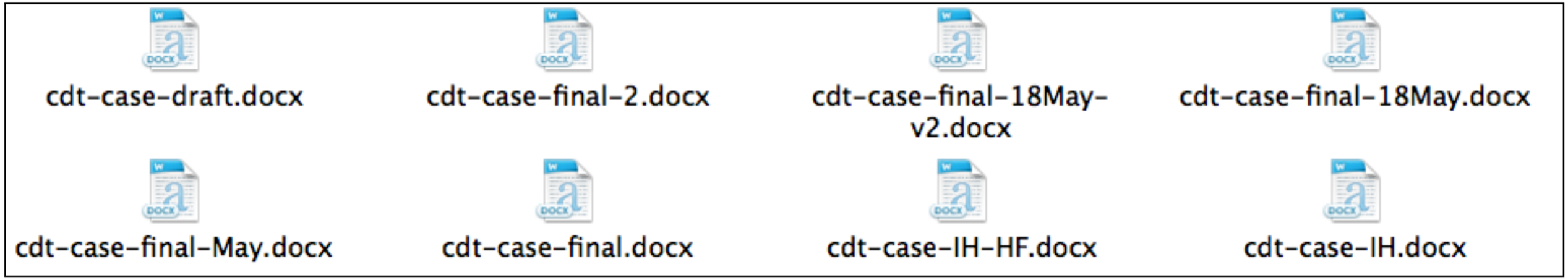
How to address these challenges?

- Technical solutions:
  - 20/80 rule applies: Embracing 20% of good practice in research software engineering, will give huge payoff.
- Policy and political solutions
  - Educate
  - Change metrics
  - Change behaviour

Essential tools for reproducible computational science

1. Version control
2. Automatic testing and
3. Continuous integration
4. Record compute environments
5. Automate everything

1. Version Control - Why?




```
gcm.f                          gcm2c-nonlin.f
gcm2.f                         gcm2c.f
gcm2b.f                        gcm2d-Dec2013.f
gcm2c-nonlin-NS.f              gcm2d.f
gcm2c-nonlin-after-AGM.f       gcm2e-Nov2013.f
gcm2f.f                        gcm-better.f
```

- More effective: to use version control tools

1. Version control - how?

- After significant changes, one 'commits' the current version files
- All commits are stored, together with time stamps and a comment. Useful information may include:
  - "as used for figure 3 in Nature paper”
  - “Have added iterative method to solver suite”
  - "implemented suggestion from reviewer A”
- All changes can be undone, compared; one can 'travel in time’
- Use for code, configuration files, scripts, documents, papers, graphs, . . .

thelibrary.py

```python
"""Collection of simulation routines"""

def calculation(x, y, z):
    """Library function that carries out some calculation"""
    return x + y + z

```

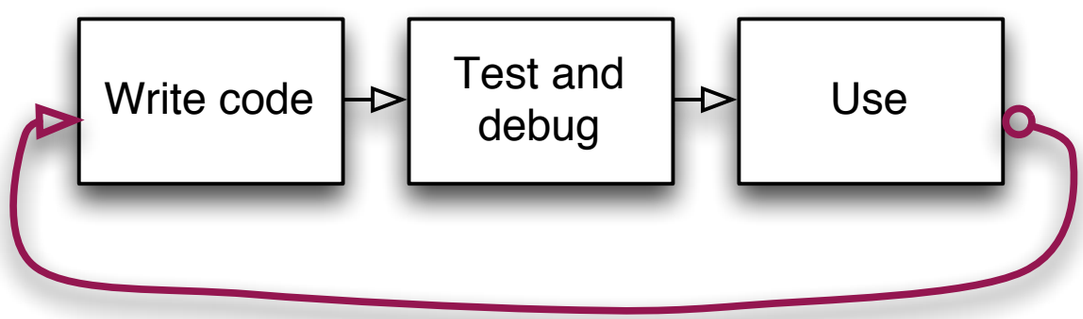
```
commit 116c7b52f70725841e219973f2f0514d082ec59d
Author: Hans Fangohr iota <fangohr@soton.ac.uk>
Date:   Wed Mar 29 12:41:44 2017 +0100

    Add library and test

 test_thelibrary.py | 15 +++++++++++++++++
 thelibrary.py      |  5 +++++
 2 files changed, 20 insertions(+)
```

Host repository on Github:
https://github.com/fangohr/demo-computational-science-essential-tools

2. Automatic tests



- Testing is crucial but time consuming
- Idea: have additional computer code that tests the actual code
- run this ‘testing code’ every time we change the main code
- known as ‘unit/system/regression/... testing’

thelibrary.py | test_thelibrary.py

```python
import thelibrary as t

def test_calculation_positive():
    assert t.calculation(1, 2, 3) == 6
    assert t.calculation(0, 20, 3) == 23

def test_calculation_negative():
    assert t.calculation(-1, -2, 3) == 0
    assert t.calculation(-1, -2, -100) == -103

def test_calculation_bordercase():
    assert t.calculation(0, 0, 0) == 0

def test_calculation_floats():
    assert t.calculation(0.5, 0.5, 1.) == 2.0

```

```
$ py.test -v test_thelibrary.p
================== test session starts ========
collected 4 items

test_thelibrary.py::test_calculation_positive PASSED
test_thelibrary.py::test_calculation_negative PASSED
test_thelibrary.py::test_calculation_bordercase PASSED
test_thelibrary.py::test_calculation_floats PASSED

========= 4 passed in 0.01 seconds =============
```

3. Continuous integration

- Run automatic tests routinely. For example:
  - After every code change
  - Every week
- Failures being recorded and reported
- Free and commercial tools and cloud-hosted services available.

https://travis-ci.org/fangohr/demo-computational-science-essential-tools

Commit 00f3558

Compare cc494dc..00f3558

Branch master

Hans Fangohr iota authored and committed

Ran for 43 sec

about 2 hours ago

Job log | View config

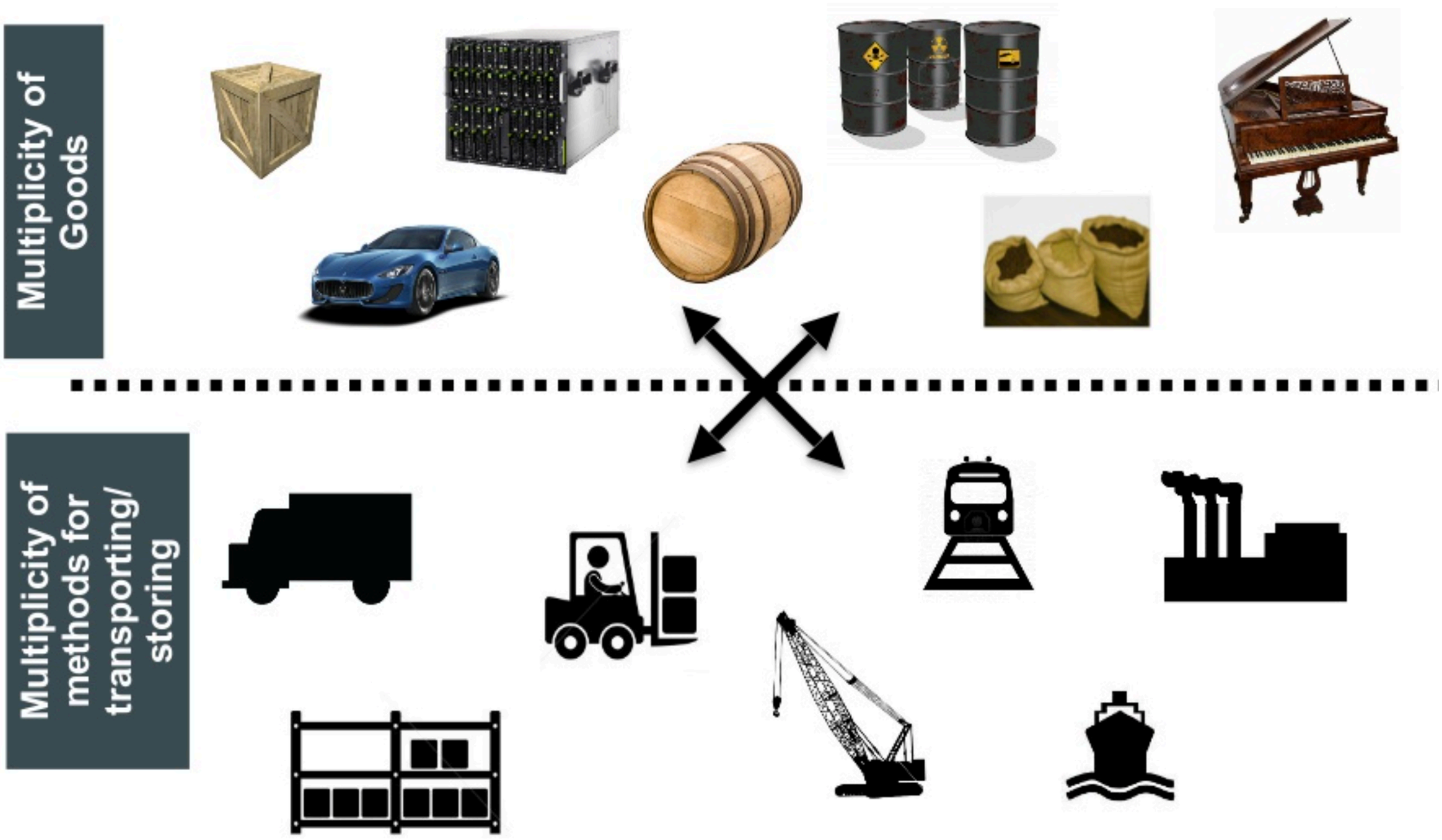
Remove log | Raw log

```
1   Worker information                                                                  worker_info
6   Build system information                                                            system_info
73
74  $ export DEBIAN_FRONTEND=noninteractive                                             fix.CVE-2015-7547
110 3.6 is not installed; attempting download
111 Downloading archive: https://s3.amazonaws.com/travis-python-archives/binaries/ubuntu/12.04/x86_64/python-3.6.tar.bz2
112 $ sudo tar xjf python-3.6.tar.bz2 --directory /
113 $ git clone --depth=50 --branch=master https://github.com/fangohr/demo-computational-science-   git.checkout   0.38s
123
124 This job is running on container-based infrastructure, which does not allow use of 'sudo', setuid and setguid executables.
125 If you require sudo, add 'sudo: required' to your .travis.yml
126 See https://docs.travis-ci.com/user/workers/container-based-infrastructure/ for details.
127 $ source ~/virtualenv/python3.6/bin/activate                                          0.01s
128
129 $ python --version
130 Python 3.6.0
131 $ pip --version
132 pip 9.0.1 from /home/travis/virtualenv/python3.6.0/lib/python3.6/site-packages (python 3.6)
133 Could not locate requirements.txt. Override the install: key in your .travis.yml to install dependencies.
134 $ make test                                                                           0.41s
135 py.test -v .
136 ============================= test session starts ==============================
137 platform linux -- Python 3.6.0, pytest-3.0.5, py-1.4.32, pluggy-0.4.0 -- /home/travis/virtualenv/python3.6.0/bin/python
138 cachedir: .cache
139 rootdir: /home/travis/build/fangohr/demo-computational-science-essential-tools, inifile:
140 collected 4 items
141
142 test_thelibrary.py::test_calculation_positive PASSED
143 test_thelibrary.py::test_calculation_negative PASSED
144 test_thelibrary.py::test_calculation_bordercase PASSED
145 test_thelibrary.py::test_calculation_floats PASSED
146
147 ========================== 4 passed in 0.02 seconds ===========================
148
```

4. Compute environment: Containers

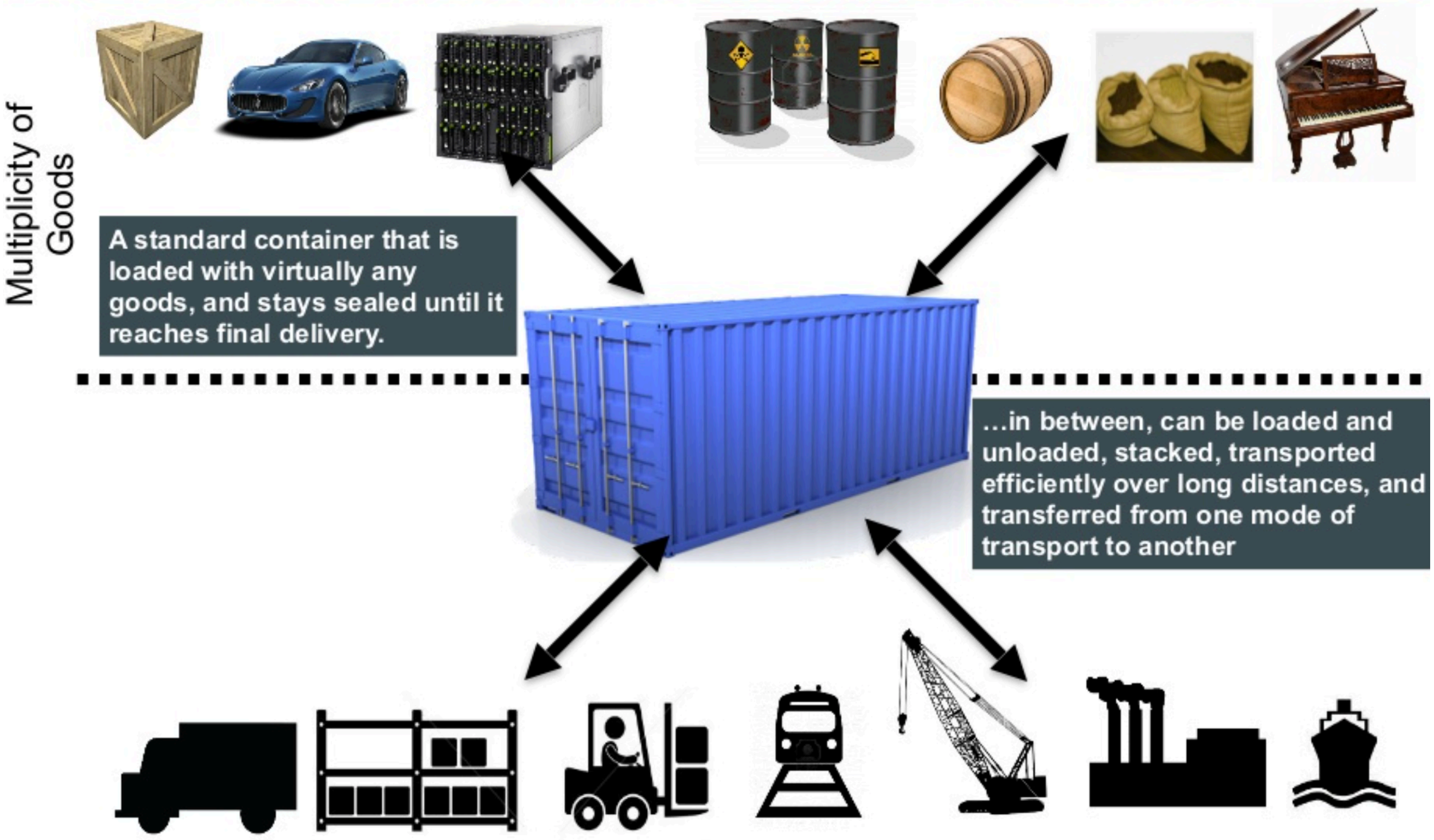
- Light weight virtual machine
- Fairly recent development
- Hot container applications at the moment:
  - Docker
  - Singularity

Cargo Transport Pre-1960



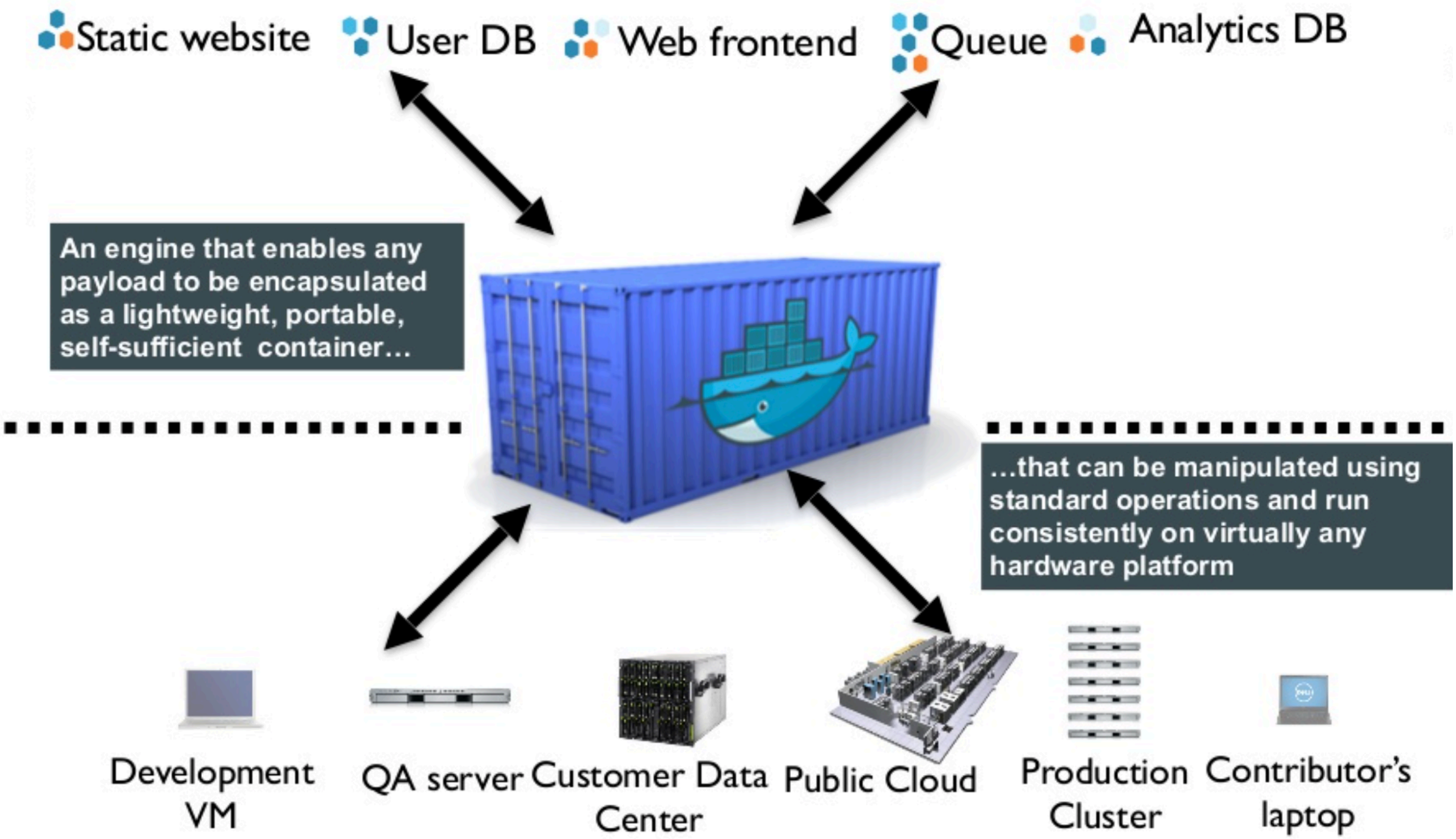
https://www.slideshare.net/jonasrosland/docker-and-containers-overview-docker-workshop

Solution: Intermodal Shipping Container



https://www.slideshare.net/jonasrosland/docker-and-containers-overview-docker-workshop

Docker is a shipping container system for code



https://www.slideshare.net/jonasrosland/docker-and-containers-overview-docker-workshop

Hello world in Container

- `$docker pull ubuntu:latest`
- `$docker run ubuntu:latest cat /etc/issue`
- `$docker run ubuntu:latest echo Hello World`

Build and use our own container

- Need Dockerfile:

```
FROM ubuntu:16.04
RUN apt-get update
RUN apt-get install -y python3
RUN mkdir /io
WORKDIR /io
```

- Create docker container

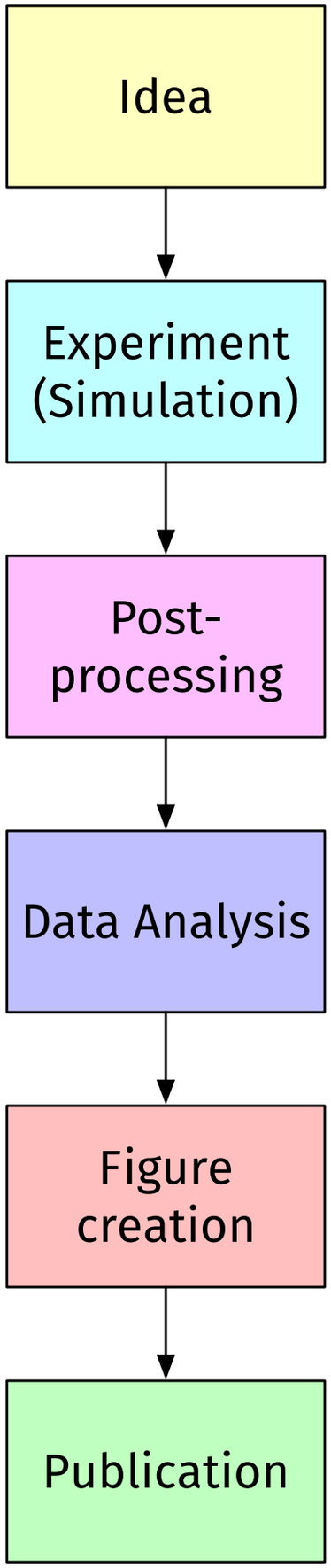
```
$ docker build -t example .
```

- Execute code in container:

```
$ docker run -v `pwd`:/io example python3 add.py
```

4. Containers Summary

- fantastic tool to document the computation environment (‘Dockerfile’ is script)
- Use cases include: installation, portability, reproducibility, testing



5. Automate everything

- Computers are great doing repetitive stuff.
- Let them do it.
- Humans should focus on tasks only humans can do.
- Avoid GUIs for **final** analysis; use scripts instead.

Essential tools for reproducible computational science

1. Version control - **keep all versions of all files safe**
2. Automatic testing - **gives confidence, saves time, enables change of the code**
3. Continuous integration - **runs tests routinely**
4. Record compute environments (for example containers) - **allows to re-run computation**
5. Automate everything - **saves time (in the long run ...), allows to repeat complete analysis by touching one button.**

SCIENTIFIC REPORTS

OPEN

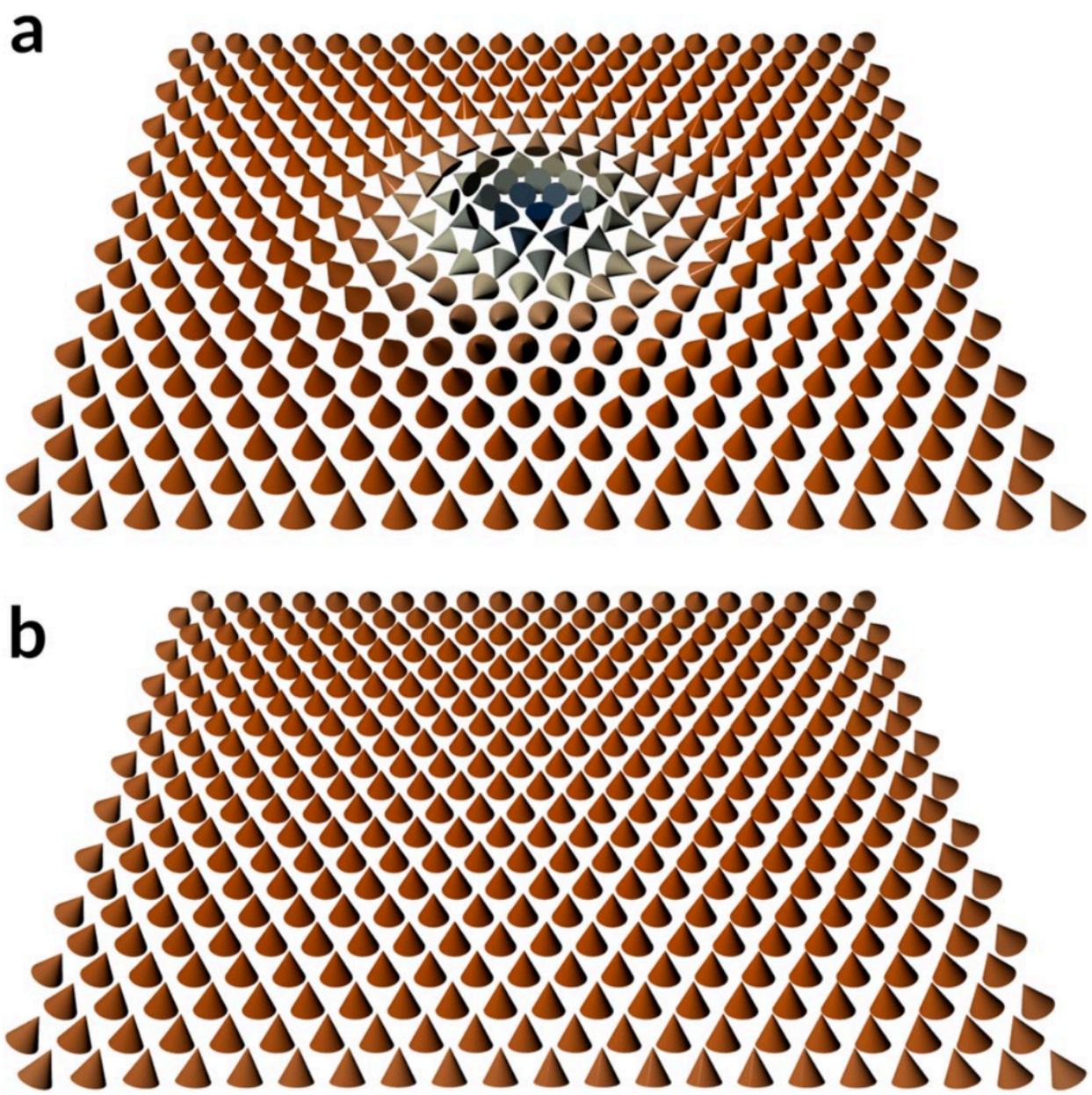
Thermal stability and topological protection of skyrmions in nanotracks

Received: 4 January 2017
Accepted: 19 April 2017
Published online: 22 June 2017

David Cortés-Ortuño[1], Weiwei Wang[1,2], Marijan Beg[1], Ryan A. Pepper[1], Marc-Antonio Bisotti[1], Rebecca Carey[1], Mark Vousden[1], Thomas Kluyver[1], Ondrej Hovorka[1] & Hans Fangohr[1,3]

Magnetic skyrmions are hailed as a potential technology for data storage and other data processing devices. However, their stability against thermal fluctuations is an open question that must be answered before skyrmion-based devices can be designed. In this work, we study paths in the energy landscape via which the transition between the skyrmion and the uniform state can occur in interfacial Dzyaloshinskii-Moriya finite-sized systems. We find three mechanisms the system can take in the process of skyrmion nucleation or destruction and identify that the transition facilitated by the boundary has a significantly lower energy barrier than the other energy paths. This clearly demonstrates the lack of the skyrmion topological protection in finite-sized magnetic systems. Overall, the energy barriers of the system under investigation are too small for storage applications at room temperature, but research into device materials, geometry and design may be able to address this.

European XFEL



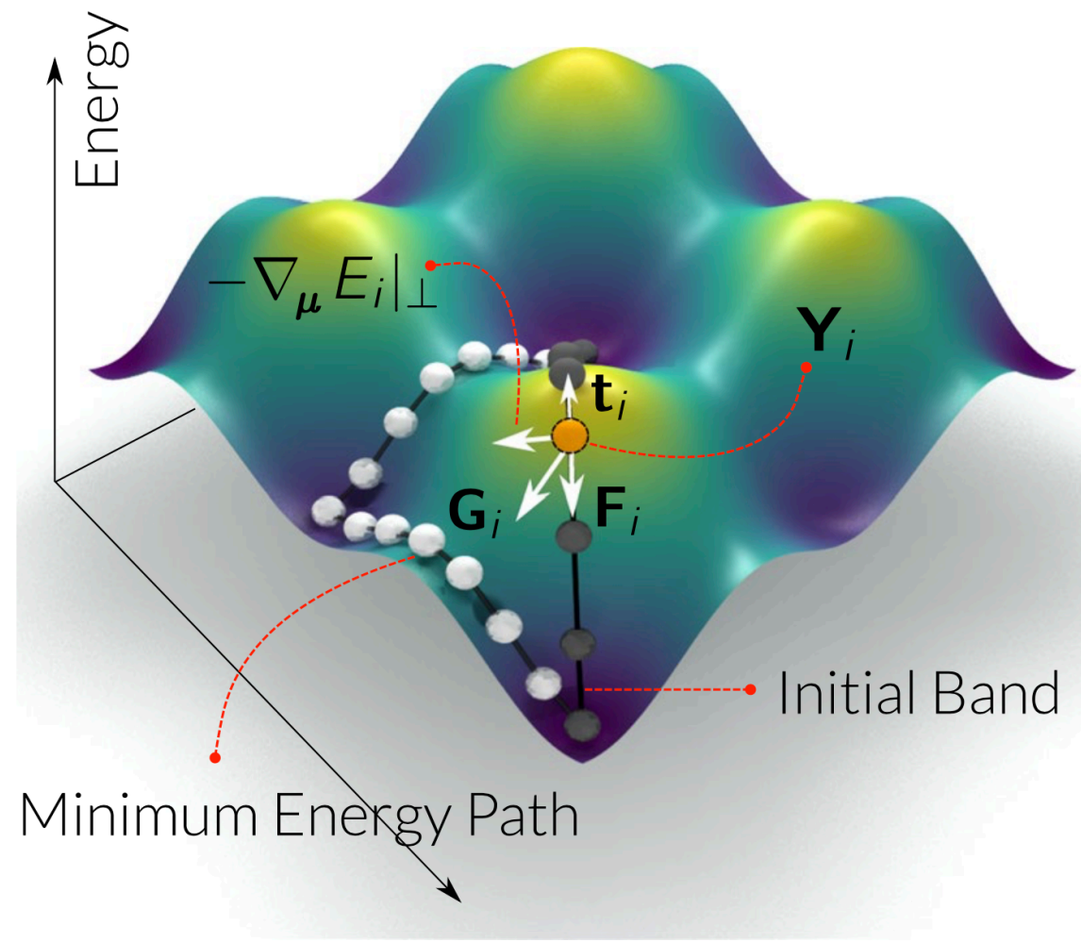


Figure 1. Magnetic configurations in a system with interfacial DMI. Representation of: (**a**) a Néel skyrmion and (**b**) a ferromagnetic ordering in a thin magnetic film with interfacial DMI.

https://github.com/davidcortesortuno/paper-2016-Cortes-etal-Thermal-stability-and-topological-protection-of-skyrmions

davidcortesortuno / paper-2016-Cortes-etal-Thermal-stability-and-topological-protection-of-skyrmions

Unwatch 2 | Star 1 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

Data for the paper about energy barriers in skyrmionic magnetic nanotracks

30 commits | 1 branch | 1 release | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

davidcortesortuno committed on GitHub Update README.md — Latest commit 458a891 on 21 Nov 2016

docker	Updated README with more details and plot examples. Updated plotting ...	4 months ago
figs	Updated README with more details and plot examples. Updated plotting ...	4 months ago
plot	Fixed plot of snapshots of bands with an odd number of images	4 months ago
sims	Fixed plot of snapshots of bands with an odd number of images	4 months ago
Makefile	Updated README with Docker based simultions instructions	4 months ago
README.md	Update README.md	4 months ago

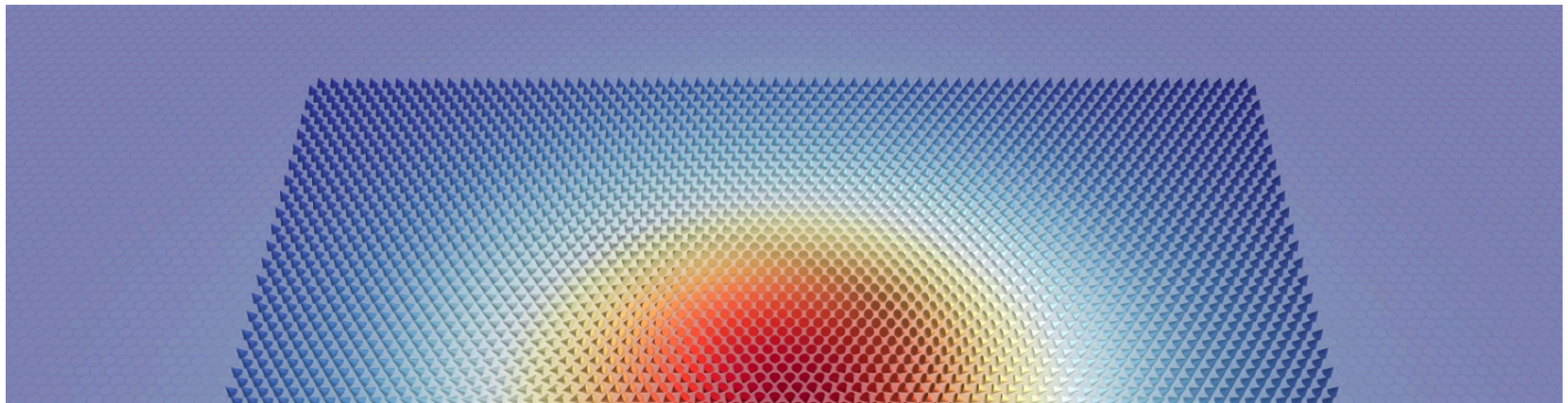
README.md

DOI 10.5281/zenodo.167874

NEBM Simulations

This repository contains all the necessary scripts to run the Nudged Elastic Band method (NEBM) atomistic simulations, using Fidimag.

System



davidcortesortuno / paper-2016-Cortes-etal-Thermal-stability-and-topological-protection-of-skyrmions

Unwatch 2 | Star 1 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

Branch: master | Find file | Copy path

paper-2016-Cortes-etal-Thermal-stability-and-topological-protection-of-skyrmions / Makefile

davidcortesortuno Updated README with Docker based simultions instructions — d99b42a on 17 Nov 2016

1 contributor

19 lines (16 sloc) | 587 Bytes — Raw | Blame | History

```
example:
        @echo "Building Docker container:"
        @cd docker && make image
        @echo "Running NEBM simulations for D = 0.721 meV (D = 3.2 mJ m **-2)"
        @cd docker && export DMI=32 && make relaxation && make nebm && make plot_nebm

run_all:
        @echo "Building Docker container:"
        @cd docker && make image
        @echo "Running NEBM simulations for different DMI values"
        @cd docker && \
                for D in 26 28 30 32 34 36; do \
                        export DMI=$$D && make relaxation && make nebm && make plot_nebm;  \
                done

clean_docker:
        @echo "Attempting to remove docker container: nebm"
        @cd docker && make clean_image
```

davidcortesortuno Changed option to clean images. Added option to plot computed energy ... 7f21768 on 16 Nov 2016

1 contributor

38 lines (29 sloc) | 1.13 KB

Raw | Blame | History

```
FROM ubuntu:16.04

RUN apt-get -y update
RUN apt-get install -y git python3 python3-pip gcc psutils cmake wget make
RUN apt-get install -y gfortran libblas-dev liblapack-dev python3-tk sudo fonts-lato
RUN pip3 install cython matplotlib pytest scipy psutil pyvtk ipywidgets

RUN ln -s /usr/bin/python3 /usr/bin/python

WORKDIR /usr/local
RUN git clone https://github.com/computationalmodelling/fidimag.git
WORKDIR /usr/local/fidimag/bin
RUN bash install-fftw.sh
RUN bash install-sundials.sh

ENV PYTHONPATH="/usr/local/fidimag:$PYTHONPATH"
ENV LD_LIBRARY_PATH="/usr/local/fidimag/local/lib:$LD_LIBRARY_PATH"

WORKDIR /usr/local/fidimag
RUN python3 setup.py build_ext --inplace
RUN python3 -c "import matplotlib"
# Headless Matplotlib:
ENV MPLBACKEND=Agg

ENV OMP_NUM_THREADS=2
WORKDIR /io

```

Useful tools

- Python
- Jupyter Notebook

Python

- Core is easy to learn [1]
- Easy to read and flexible => high productivity language
- Not fast if used naively
- Popularity in science reflects the high cost of the programmer (=scientist)

[1] Hans Fangohr: A Comparison of C, Matlab and Python as Teaching Languages in Engineering, Lecture Notes on Computational Science 3039, 1210-1217 (2004), https://eprints.soton.ac.uk/22811/

“Embedded Domain specific language (DSL)”

```
import thelibrary

# Important parameters
a = 41.2
b = 0.7
c = 0.1

# call simulation with important parameters
results = thelibrary.calculation(a, b, c)

# write result to disk, or here for simplicity to stdout
print(results)
```

- “Configuration file” is general purpose program

Marijan Beg, Ryan A. Pepper, Hans Fangohr: *User interfaces for computational science: a domain specific language for OOMMF embedded in Python*, AIP Advances 7, 056025 (2017), https://arxiv.org/abs/1609.07432

Workflow tools: Jupyter Notebook

- Originally “IPython Notebook”
- > 2 million users (in 2015)
- web-based interactive computing platform
- Combine code, output, interpretation in executable document

jupyter jupyterdemo1 (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

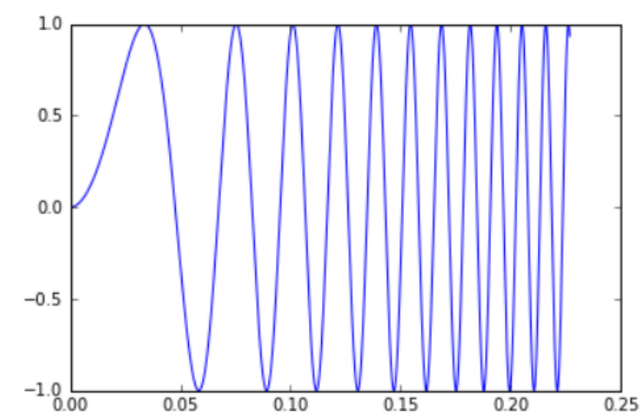
In [1]:
```
1 + 2
```
Out[1]: 3

Cells can contain text and latex equations such as $f(x) = \sin(2\pi\omega t^2)$ and $\omega = 220$ Hz. We can use code to define the corresponding functions:

In [2]:
```
import numpy as np
def f(t):
    omega = 220
    return np.sin(2 * np.pi * omega * t**2)
```

Let's compute the data and plot the beginning of it:

In [5]:
```
t = np.linspace(0, 2, 44100)
y = f(t)
## Show plots inside the notebook
%matplotlib inline
import pylab
pylab.plot(t[0:5000], y[0:5000])
```
Out[5]:
```
[<matplotlib.lines.Line2D at 0x10f7be940>]
```



We can integrate media: images, videos, interactive elements and sound:

In [4]:
```
from IPython.display import Audio
Audio(y, rate=44100)    # plays the data in y as audible signal
```
Out[4]:

We can connect other languages and tools, for example execution in bash:

In [6]:
```
%%bash
echo "Some shell command, run at `date`"
```
```
Some shell command, run at Tue  9 Aug 2016 14:19:57 BST
```

In [ ]:

jupyter notebook-demo Last Checkpoint: 15 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

Title of investigation

Want to understand $f(t) = \exp(-\alpha t)\cos(\omega t)$

In [ ]:
```
def f(t, alpha, omega):
    """Computes and returns exp(-alpha*t) * cos(omega*t)"""
    return exp(-alpha * t) * cos(omega * t)
```

We can execute the function for valuel of α and ω:

In [ ]:
```
f(t=0.1, alpha=1, omega=10)
```

Although sometimes a plot is more instructive:

In [ ]:
```
def plot_f(alpha, omega):
    ts = linspace(0, 5, 400)    # many points in the interval [0, 5]
    ys = f(ts, alpha, omega)
    pylab.plot(ts, ys, '-')
```

In [ ]:
```
plot_f(alpha=0.1, omega=12)
```

A wide range of convenience tools, for example graphical interaction elements called *Widgets*:

In [ ]:
```
interact(plot_f, alpha=(0, 2, 0.1), omega=(0, 20, 0.5))
```

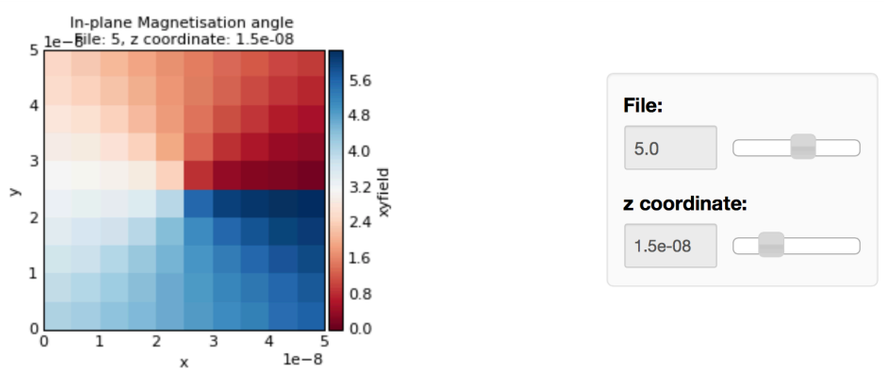
Conclusion

So we observe: Parameter α is responsible for damping, and ω for the frequency.

- Notebook allows bitmap/ svg/html... representations
- more user friendly interfaces
- LaTeXed equations (right)
- Images (right)
- Interactive “widgets” (below)

In [16]:
```
#PYTEST_VALIDATE_IGNORE_OUTPUT
inplane = joommftools.create_inplane_holomap(files[:10], slicecoords, 'z')
inplane
```

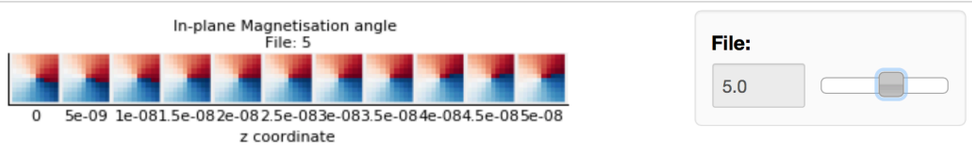
Out[16]:



These maps can be aggregated into grids by grouping over given dimensions, to produce static plots. This is useful for visualising multimensional data, and is often used in phase diagrams in micromagnetic publications. Here we grid the Holomap which shows us how each file along it's z axis:

In [17]:
```
#PYTEST_VALIDATE_IGNORE_OUTPUT
inplane.grid(['z coordinate'])
```

Out[17]:

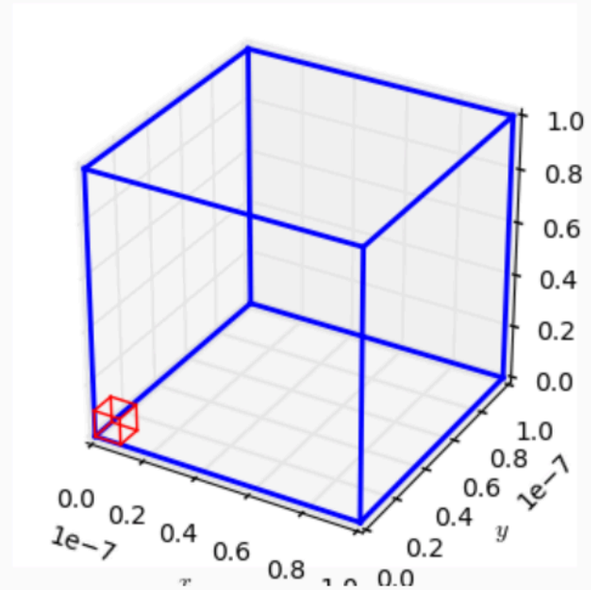


We initialise the system in (0, 0, 1) direction, which is clearly different from our expected equlibrium state.

In [16]:
```
system.m = df.Field(system.mesh, value=(0, 1, 0), normalisedto=8e6)
```

We can check the characteristics of the system we defined

In [17]:
```
%matplotlib inline
system.mesh
```



In [18]:
```
system.hamiltonian
```

Out[18]:

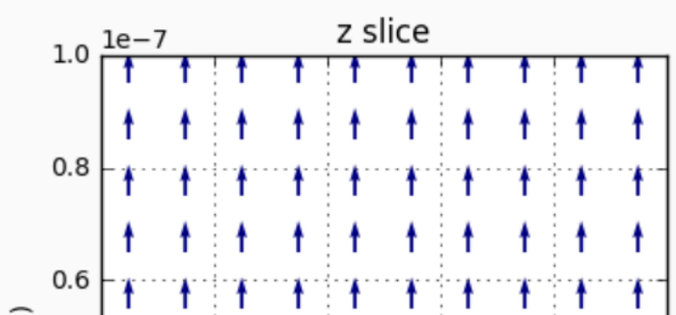
$$\mathcal{H} = A[(\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2] - \frac{1}{2}\mu_0 M_s \mathbf{m} \cdot \mathbf{H}_d - \mu_0 M_s \mathbf{m} \cdot \mathbf{H}$$

In [19]:
```
system.dynamics
```

Out[19]:

$$\frac{\partial \mathbf{m}}{\partial t} = -\gamma \mathbf{m} \times \mathbf{H}_{\text{eff}} + \alpha \mathbf{m} \times \frac{\partial \mathbf{m}}{\partial t}$$

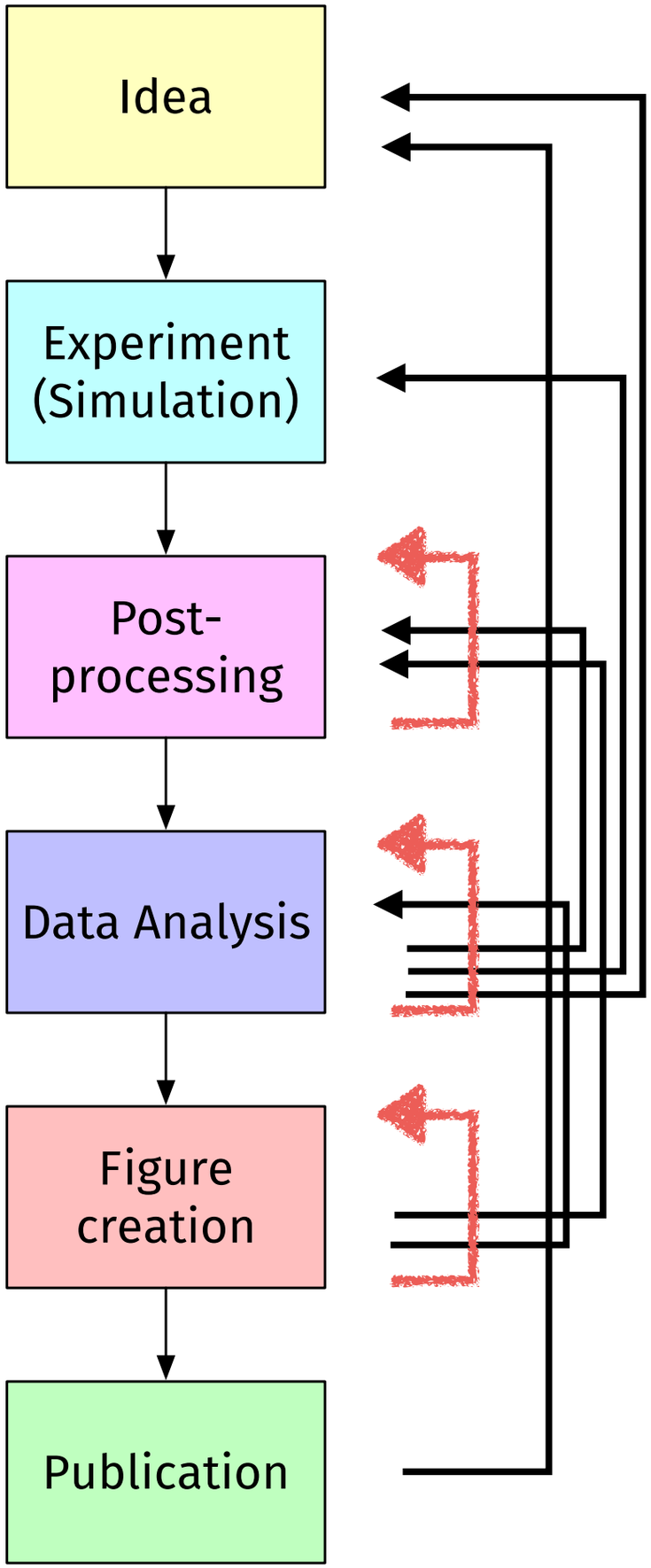
In [20]:
```
fig = system.m.plot_slice("z", 50e-9, xsize=4)
```



Jupyter notebook use cases

- Computational exploration (both data and Simulation driven)
- Documentation [1]
- Reproducible computation [2]
- Teaching & Learning [3]
- Communicating studies to supervisors, collaborators, …

[1] http://oommfc.readthedocs.io/en/latest/ipynb/standard_problem3.html & https://github.com/joommf/oommfc/blob/master/docs/ipynb/standard_problem3.ipynb
[2] https://github.com/maxalbert/paper-supplement-nanoparticle-sensing
[3] https://github.com/fangohr/introduction-to-python-for-computational-science-and-engineering



Scientific Workflow

- Iterative exploration of a problem/data set via computation and intermediate results
- At the end of the study, results must be communicated through a (linear) *narrative* (paper/report/thesis,...).
- Should be reproducible.
- => Notebook delivers this “for free”

Challenges for software in science

- reproducibility
- testing can be difficult
- ongoing changes throug
- no incentive to documer
- coders may not have pro
- main code author often
- code lost or not publicly
- fast execution competes
- hardware changes require re-write of codes

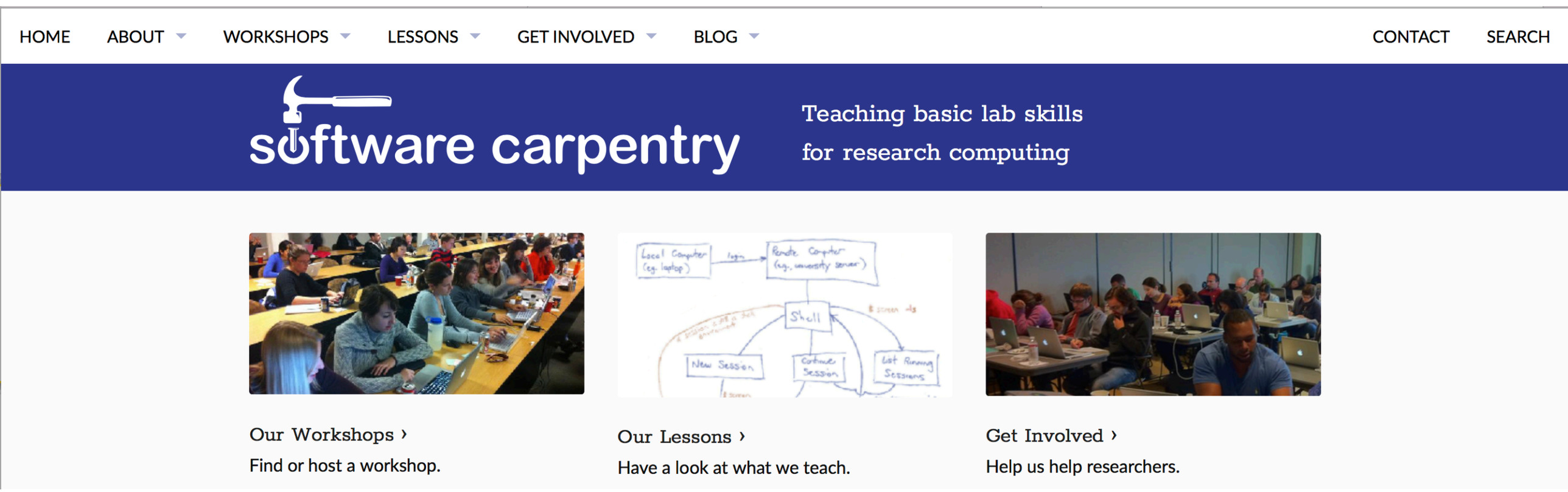
- Technical solutions
  - Automate everything
  - Version control
  - Automatic tests &
  - Continuous integration
  - Compute environment
  - Python
  - Jupyter Notebook

Other measures

- Training
  - Software Carpentry
  - Doctoral Training centre
- Policy and Politics
  - Software as infrastructure
  - Research software engineer
  - Software sustainability institute

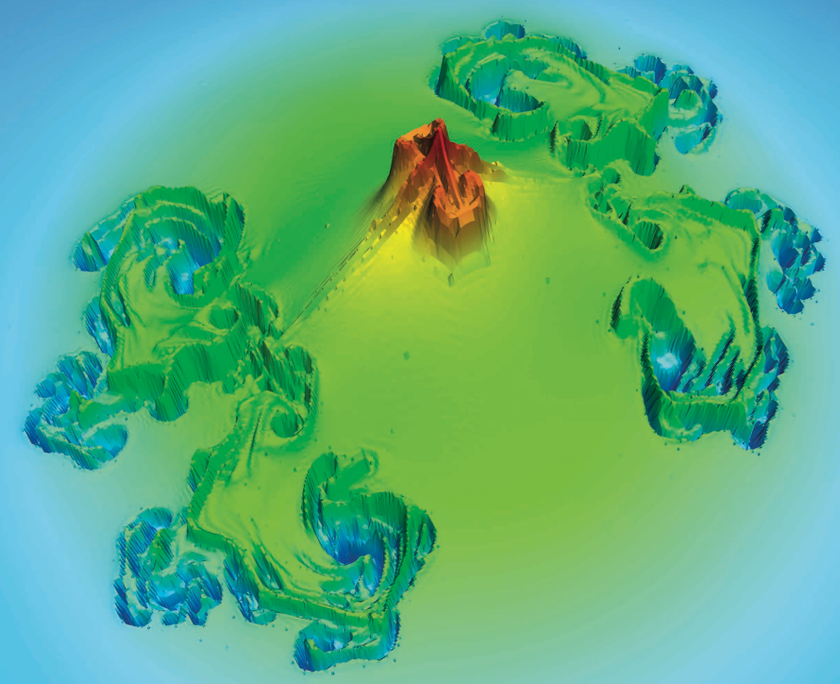
Software Carpentry

- software-carpentry.org
- Volunteers delivering basic computational training for scientist, for example:
  git, bash, python, tests, R, MySQL, ...



4-year Doctoral Training Programme

- 1-year Masters + 3 year PhD
- Best practice computational science, including software engineering for science & data science
- 12 million EUR investment by UK research council, industry partners and University of Southampton
- 75 PhD students starting over 5 years
- Change making "**from bottom up**"



Policies and politics

- Research Councils (in UK) start to accept **Software as an infrastructure investment**
  - Focus of investment (UK Scientific Advisory Committee on High Performance Computing)
  - Funding for Research Software Engineer fellowships
- Emerging Job profile of "**Research Software Engineer**" (in addition to post-doc and academic)
- Data Management plan (-> Software Management plan?)

- Information, Networking, Fellowships
- United Kindgom Research council funded
- https://www.software.ac.uk
- Similar initiatives starting in Germany

Summary

- Challenges for Computational Science
- Need to improve software quality
- Introduced some options
  - Technical
  - Politics, education & training [1]

Hans Fangohr
http://fangohr.github.io
hans.fangohr@xfel.eu
@ProfCompMod

[1] http://www.soton.ac.uk/~fangohr/teaching

Thank you for your attention!

Hans Fangohr
http://fangohr.github.io
hans.fangohr@xfel.eu
@ProfCompMod

European XFEL

What is so hard about this ?

Context & trivial example:

Adding *n* numbers

Trivial example

- Let's pretend our scientific task is to add up three numbers, say 41.2, 0.1 and 0.7.
- Possible solutions include
  - C++ program
  - Python program
  - Excel spreadsheet
  - Calculator on smart phone
  - Data base

C++

add.cpp

```cpp
#include <iostream>

int main()
{
    double a, b, c, s;

    a = 41.2;
    b = 0.1;
    c = 0.7;

    s = a + b + c;

    std::cout << s << std::endl;;

    return 0;
}
```

```
42
[Finished in 1.0s]
```

C++

add.cpp

```cpp
#include <iostream>

int main()
{
    double a, b, c, s;

    a = 41.2;
    b = 0.1;
    c = 0.7;

    s = a + b + c;

    std::cout << s << std::endl;;

   return 0;
}
```

```
42
[Finished in 1.0s]
```

Python

add.py

```python
a = 41.2
b = 0.1
c = 0.7
s = a + b + c
print(s)
```

```
42.00000000000001
[Finished in 0.1s]
```

C++

add.cpp

```cpp
#include <iostream>

int main()
{
    double a, b, c, s;

    a = 41.2;
    b = 0.1;
    c = 0.7;

    s = a + b + c;

    std::cout << s << std::endl;;

   return 0;
}
```

```
42
[Finished in 1.0s]
```

C++

add2.cpp

```cpp
#include <iostream>
#include <iomanip>

int main()
{
    double a, b, c, s;

    a = 41.2;
    b = 0.1;
    c = 0.7;

    s = a + b + c;

    std::cout << std::setprecision(20) << s << std

   return 0;
}
```

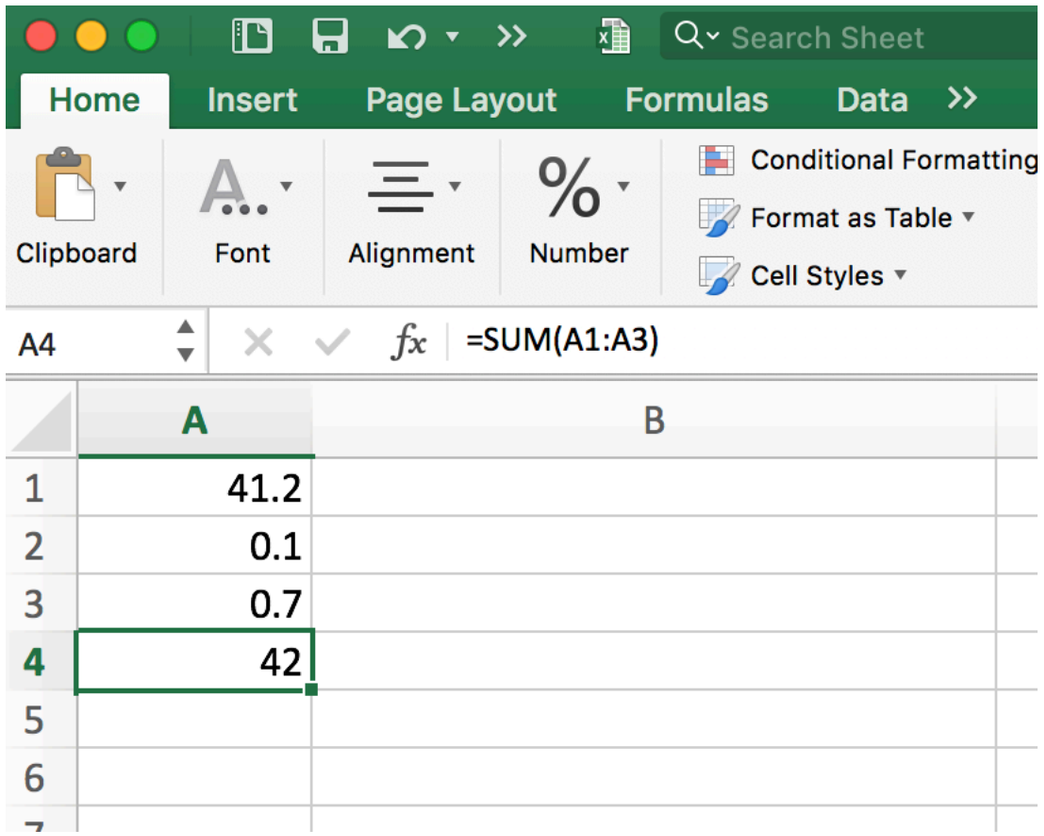
```
42.000000000000007105
[Finished in 0.3s]
```

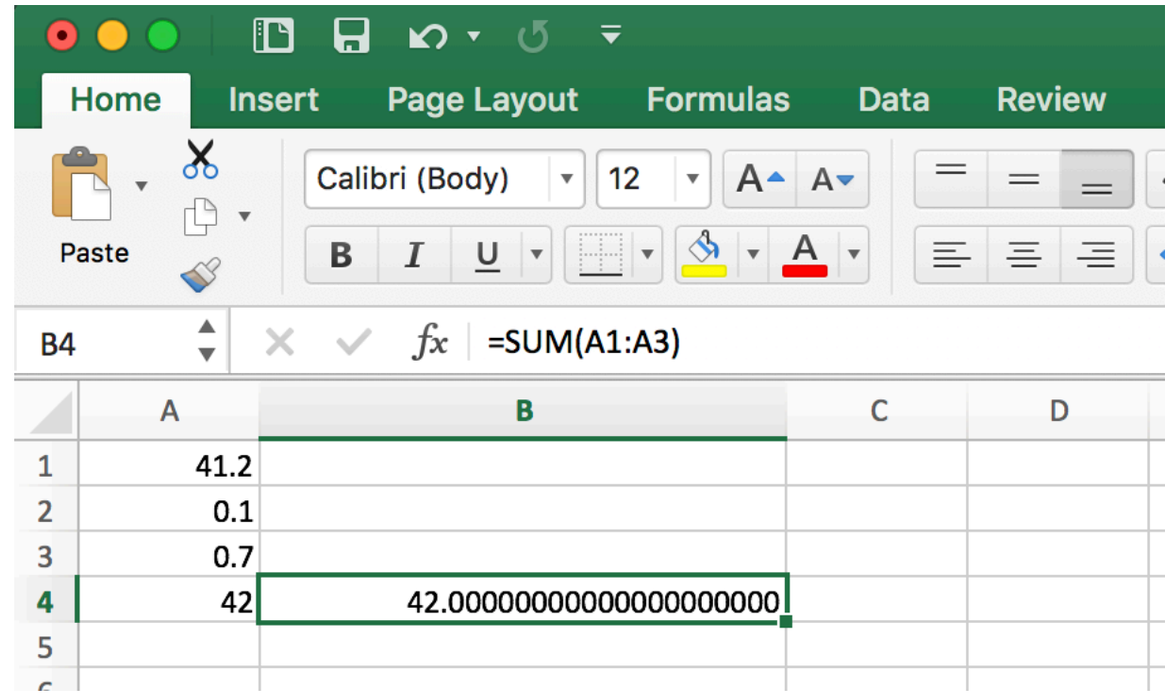
Line 1, Column 1 Tab Size: 4 C++

Issues with simple example

- `41.2 + 0.1 + 0.7 should be 42.0 but`
- `41.2 + 0.1 + 0.7 == 42.000000000000007105`
  - `Details:`
  - `a = 41.2  ≈ 41.200000000000002842`
  - `b =  0.1  ≈  0.100000000000000006`
  - `c =  0.7  ≈  0.699999999999999956`
  - ———————————————————————————
  - `a + b + c = 42.000000000000007105`

MS Excel





In Microsoft Excel, the answer is always 42.0:

- It does something (superficially) clever [1]

[1] https://support.microsoft.com/en-us/help/78113/floating-point-arithmetic-may-give-inaccurate-results-in-excel

Actually, it is worse . . .

```
a = 41.2  ≈ 41.200000000000002842
b =  0.1  ≈  0.100000000000000006
c =  0.7  ≈  0.699999999999999956
---------------------------------
a + b + c = 42.000000000000007105
c + b + a = 42.000000000000000000
```

-> Addition is not commutative (the order matters)!

[Tested with Python 3.6 and Apple LLVM version 8.1.0 / clang-802.0.42]

Design questions / metrics

- Scaling for more data - what if $N=10^9$? Execution performance?
- Re-usability of code? Separate code and data?
- Testability?
- Portability across operating systems? Architectures?
- Accessibility? - can people with different different computer/ OS use the file?
- Reproducibility?
- Readability by non-experts - is the ‘code’ understandable?
- What if we want to carry out the operation symbolically?
- Open Source?
- …

Possible solution

thelibrary.py

```python
"""Collection of simulation routines"""

def calculation(x, y, z):
    """Library function that carries out some calculation"""
    return x + y + z

```

task.py

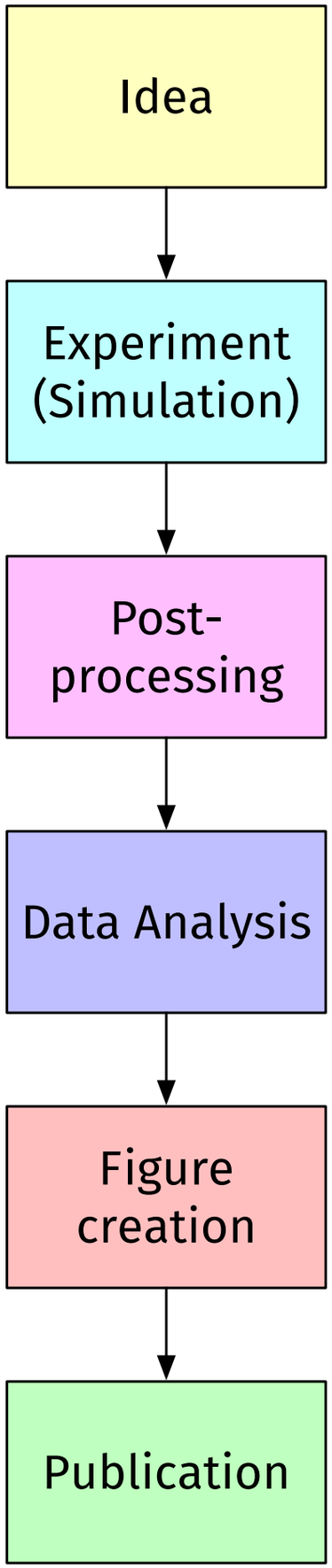
```python
import thelibrary

# Important parameters
a = 41.2
b = 0.7
c = 0.1

# call simulation with important parameters
results = thelibrary.calculation(a, b, c)

# write result to disk, or here for simplicity to stdout
print(results)

```



European XFEL Example

- Apply detector calibration to raw data on the fly.
- Allows to change detector calibration later, improve analysis and results.
- Must be able to re-run analysis chain with new calibration to see effect.
- Prerequisite: must be able to re-run ("reproduce") previous analysis without change in final result.