Universität Rostock
Traditio et Innovatio

Sascha Spors[1], Hagen Wierstorff[2], Matthias Geier[1], Fiete Winter[1], Nara Hahn[1], Frank Schultz[1]
[1]Institute of Communications Engineering, University of Rostock, Germany
{fiete.winter; matthias.geier; nara.hahn; frank.schultz; sascha.spors}@uni-rostock.de
[2]audEERING GmbH
hagenw@posteo.de

Software Tools and Workflows for Open Science

Introduction

Science relies on the traceability and replicability of results. Ideally the whole process, from initial theoretical and / or empirical work, up to publishing the outcome, shall be performed under the open science and research paradigm.

Recent efforts in the open source software community led to convenient tools and concepts applicable to research data management. Standard approaches such as LaTeX for typesetting, Git as versioning control system or more recently the programming language Python with its myriad of scientific libraries have been well established. Currently many other concepts evolve as common tools for open science, such as automated documentation parsers and online compilers.

In this contribution we discuss and demonstrate a reliable workflow for open science/research starting from initial ideas to publishing results based on the open source IT tools.

Open Science and Open Education

Reproducible science is the gold standard for sustainable research. This requires the concept and mindset of openness[1], such as

- Open Science
- Open Methodology
- Open Source
- Open Data
- Open Peer Review
- Open Access
- Open Education.

These efforts are accompanied by the Budapest Open Access Initiative[2] started in 2002 and the Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities[3] in 2003.

Open science and open education should generally follow the same working principles as they do not differ much for its intended outcome. Starting from an initial motivation, such as

- exploring new things by analytic and/or empirical research (Science) and
- exploring new didactic concepts for teaching knowledge (Education),

the following tools and concepts can be utilized to establish a workflow of reproducible science and education.

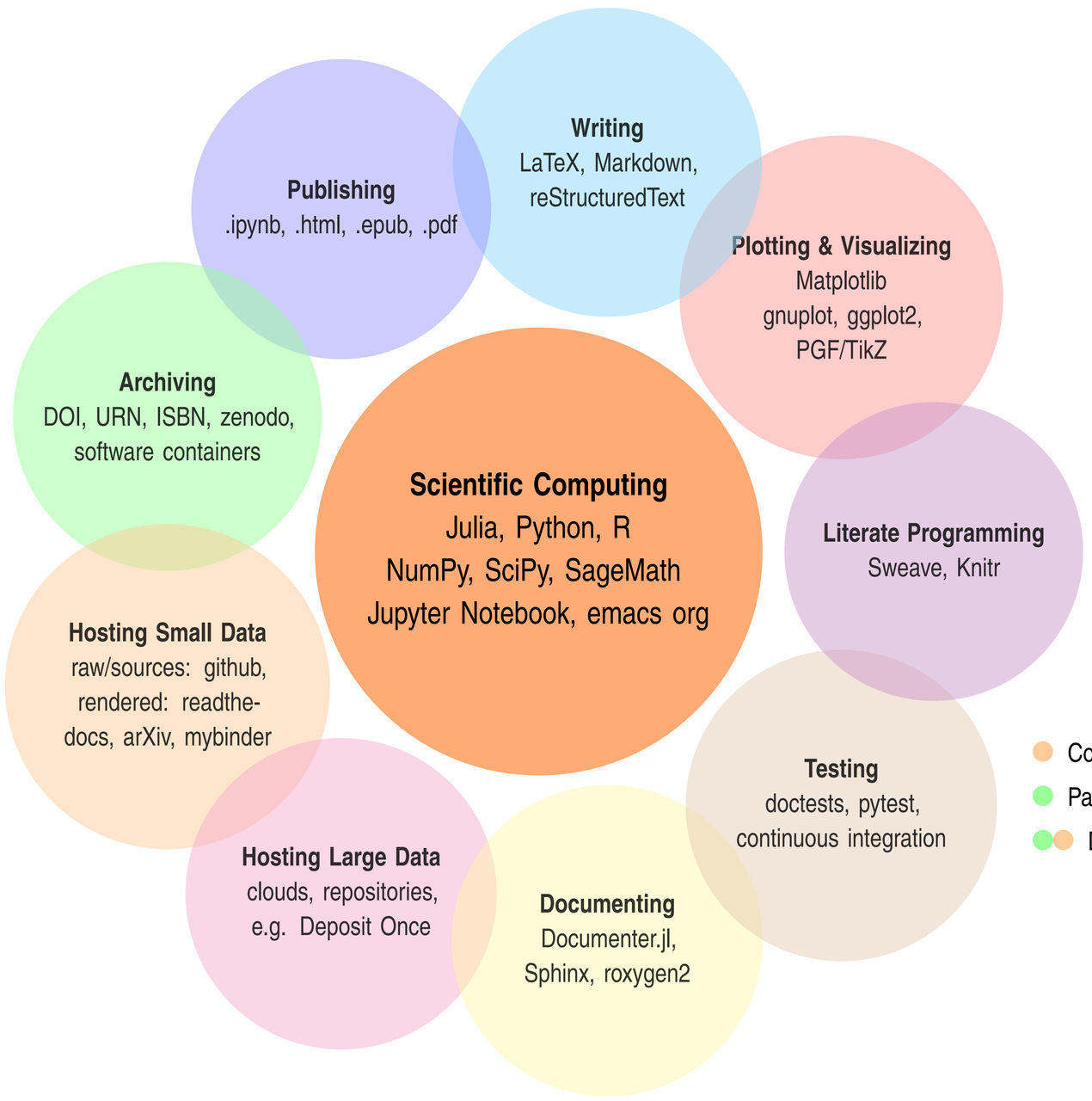
[1] https://opendefinition.org/

[2] https://www.budapestopenaccessinitiative.org/

[3] https://openaccess.mpg.de/Berlin-Declaration

Software Tools and Concepts

Everything Under Version Control (e.g. Git, Mercurial):



Workflow Example: SFS Toolbox for Python

Python code in SFS Toolbox with a docstring including LaTeX math

```python
def preeq_25d(omega, omalias, c):
    r"""Pre-equalization filter for 2.5-dimensional WFS.

    Parameters
    ----------
    omega : float
        Angular frequency.
    omalias: float
        Angular frequency where spatial aliasing becomes prominent.
    c : float
        Speed of sound.

    Returns
    -------
    float
        Complex weight for given angular frequency.

    Notes
    -----
    .. math::

        H(\w) = \begin{cases}
            \sqrt{\i \wc} & \text{for } \w \leq \w_\text{alias} \\
            \sqrt{\i \frac{\w_\text{alias}}{c}} &
            \text{for } \w > \w_\text{alias}
            \end{cases}

    """
    if omalias is None:
        return np.sqrt(1j * util.wavenumber(omega, c))
    else:
        if omega <= omalias:
            return np.sqrt(1j * util.wavenumber(omega, c))
        else:
            return np.sqrt(1j * util.wavenumber(omalias, c))
```

SFS Toolbox documentation as HTML output. Created by sphinx from the above docstring

sfs.fd.wfs.preeq_25d(*omega*, *omalias*, *c*) [source]

Pre-equalization filter for 2.5-dimensional WFS.

Parameters:
- **omega** (*float*) – Angular frequency.
- **omalias** (*float*) – Angular frequency where spatial aliasing becomes prominent.
- **c** (*float*) – Speed of sound.

Returns: *float* – Complex weight for given angular frequency.

Notes

$$H(\omega) = \begin{cases} \sqrt{\mathrm{i}\frac{\omega}{c}} & \text{for } \omega \leq \omega_\text{alias} \\ \sqrt{\mathrm{i}\frac{\omega_\text{alias}}{c}} & \text{for } \omega > \omega_\text{alias} \end{cases}$$

SFS Toolbox used within an interactive Jupyter Notebook

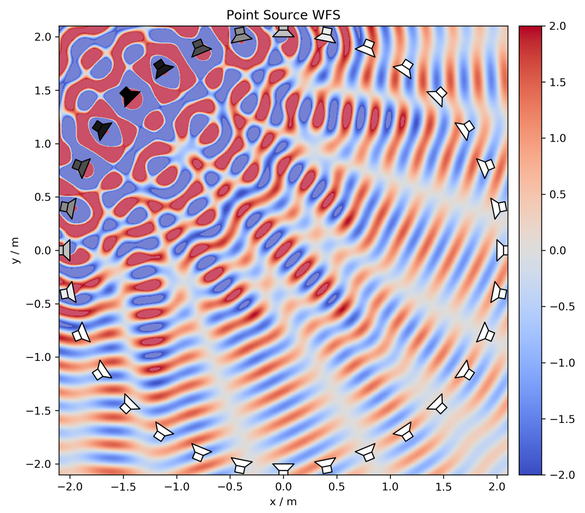
In [1]:
```python
import matplotlib.pyplot as plt
import numpy as np
import sfs
```

In [2]:
```python
R = 2  # SSD radius in m
N = 2**5  # number of sec src
ssd = sfs.array.circular(N, R)
grid = sfs.util.xyz_grid([-2.1, 2.1], [-2.1, 2.1], 0, spacing=0.005)

# primary point source
xs = -3, +3, 0  # position
As = np.linalg.norm(xs) * 4 * np.pi  # amplitude

f = sfs.default.c * 4  # frequency in Hz
omega = 2 * np.pi * f  # rad/s
print('frequency:', f, 'Hz, wavelength:', sfs.default.c / f, 'm')
```

```
frequency: 1372 Hz, wavelength: 0.25 m
```

In [3]:
```python
d, sec_src_sel, sec_src_atf = sfs.fd.wfs.point_25d(omega, ssd.x, ssd.n, xs)
p = sfs.fd.synthesize(As * d, sec_src_sel, ssd, sec_src_atf, grid=grid)
```

In [4]:
```python
sfs.plot2d.amplitude(p, grid)
ls_lvl = np.abs(ssd.a * d * sec_src_sel)
ls_lvl = ls_lvl / np.max(ls_lvl)
sfs.plot2d.loudspeakers(ssd.x, ssd.n, ls_lvl, size=0.2)
plt.title('Point Source WFS');
```



- Code hosted at https://github.com/sfstoolbox/sfs-python/
- Package hosted at https://pypi.org/project/sfs/
- Documentation hosted at https://sfs-python.readthedocs.io/

Version	Date	DOI via zenodo.org
0.5.0	2019-03-18	10.5281/zenodo.2597119
0.4.0	2018-03-14	10.5281/zenodo.1198204
0.3.1	2016-04-08	10.5281/zenodo.49356
0.3.0	2016-04-08	10.5281/zenodo.49342
0.0.0	2015-09-22	NA