DOI: 10.61782/fa.2025.0658

FORUM ACUSTICUM EURONOISE 2025

OPEN EDUCATIONAL RESOURCES FOR ACOUSTICS AND AUDIO SIGNAL PROCESSING USING JUPYTER NOTEBOOKS AND PYFAR

Fabian Brinkmann[1*] **Marco Berzborn**[2,9] **Anne Heimes**[2]
Anton Hoyer[1] **Xenofon Karakonstantis**[3] **Simon Kersten**[2]
Tim Lübeck[4] **Pascal Palenda**[2] **Cristobal Andrade**[5]
Johannes M. Arend[6] **Nara Hahn**[7] **Tobias Jüterbock**[1]
Nils Meyer-Kahlen[6] **Artur Paskiewicz**[1] **Sebastian J. Schlecht**[5]
Frank Schultz[8] **Stefan Weinzierl**[1]

[1] Audio Communication Group, Technische Universität Berlin
[2] Institute for Hearing Technology and Acoustics, RWTH Aachen University
[3] Audiological Solutions, Oticon, Smørum
[4] Institute of Computer and Communication Technology, TH Köln - University of Applied Sciences
[5] Multimedia Communications and Signal Processing, Friedrich-Alexander-Universität Erlangen
[6] Acoustics Lab, Department of Information and Communications Engineering, Aalto University
[7] Institute of Sound and Vibration Research, University of Southampton
[8] Institute of Communications Engineering, Universität Rostock
[9] German Institutes of Textile and Fiber Research, Denkendorf

ABSTRACT

Many universities offer courses on acoustics and audio signal processing, often with overlapping content. Nevertheless, the lectures and teaching material have typically been developed independently. We present a growing collection of programming exercises in the form of Jupyter notebooks that allow to interactively write and execute code including in-line graphical output such as figures. They can also contain text, equations, and multimedia content. Solutions can optionally be checked automatically using the `nbgrader` package, making notebooks a suitable tool for code-related assignments. Our notebooks are hosted on pyfar.org and are often based on the Python packages for acoustics research (pyfar). Pyfar provides functions for signal processing tasks, as well as for reading, writing, and visualizing audio data. By providing such functionality, it allows the content of the notebooks to focus on core concepts and didactics. Solutions to all exercises will be made available to teachers upon request. We hope that this initiative will foster cross-university collaboration, reduce the workload for educators, improve the quality of assignments through peer review, harmonize the content and didactics of courses across universities, and increase the number of available assignments. We will present sample assignments, the infrastructure behind the collection, and how to contribute.

Keywords: (virtual) acoustics, audio signal processing, open educational resources, python, jupyter notebook

1. INTRODUCTION

Academic teaching is integral to scientific and societal progress because it supports to educate the next generation of researchers and developers for universities and companies. Thus, the quality of education in terms of didactics and content is crucial. However, the conditions for achiev-

***Corresponding author**: *info@pyfar.org.*

ing this goal at the university level are often far from optimal. Depending on the country and the university, newly assigned lecturers frequently must develop their lectures and teaching materials under time pressure due to parallel tasks, such as setting up a new research group and raising funds for future projects. Doctoral students and postdoctoral researchers are also often involved in teaching but face their own challenges—first and foremost the need to complete their scientific work before the end of their mostly fixed-term contracts. In addition, undergraduate students involved in teaching are still in the process of internalizing the concepts and topics being taught and are therefore qualified for limited conceptual contributions.

Cross-university efforts to collaborate on teaching materials could address some of these challenges and potentially improve teaching quality through a collaborative peer review process. Although collaboration and peer review are common in research, we are aware of only a few examples in teaching. One example is the Massive Open Online Course (MOOC) on communication acoustics organized by T9, the nine technical universities in Germany [1,2]. However, this course is not free of charge. A second example is the Acoustics Courseware (acoucou) project [3], which is currently being developed by educational and industrial partners. It aims to provide open educational materials for different levels of expertise. Further, in the field of signal processing and communication acoustics, Spors et al. [4] discussed the open education resources workflow for their courses [1]. Some of these courses are heavily forked. Peer reviews are incorporated, but mostly only within the group of the initial developers. Active outside collaboration is on a very low level, currently.

Integrating programming exercises into lectures is generally advantageous, as it is an important method in engineering or more general in all algorithmic- and data-related tasks. An explicit and dedicated training on this skill will help students to cope with their future challenges in the natural sciences and engineering. Rather than introducing algorithms only in formal notation, actively implementing these often facilitates a deeper understanding of the core concepts behind the particular methods. Hence, solving analytical problems shall be accompanied by solving problems through implementation as this equivalently trains these different skills. Thus, the integration of programming assignments into lectures follows *constructive alignment*, which aims at an "alignment between the intended learning outcomes of the course, the teaching/learning activities and the assessment tasks" [5, p. 95]. At the time of writing, none of the above mentioned projects offers programming assignments in Python that could be integrated into existing or projected lectures. In the following, we propose our initiative to provide such material. By that, we aim at supporting and improving educational collaboration and material.

[1] `https://github.com/spatialaudio/signals-and-systems-exercises`

2. PYFAR OPEN EDUCATIONAL RESOURCES

The pyfar open educational resources, freely available at `pyfar.org`, provide Python-based programming assignments in the form of Jupyter notebooks [2] developed by different institutes from different countries. In the following, we introduce the scope and concepts of the pyfar open educational resources. For detailed technical instructions on how to obtain and contribute assignments, we refer the reader to the open educational resources and contributing section at `pyfar.org`.

2.1 Scope

The pyfar open educational resources are intended to provide programming assignments for acoustics and audio signal processing. At the time of writing, the majority of the assignments focused on virtual acoustics covering topics such as the

- creation of virtual sources using using head-related transfer functions,
- estimation of interaural cues from head-related transfer functions,
- interpolation of head-related transfer functions using local and global interpolation approaches,
- creation of virtual moving sources using vector base amplitude panning,
- creation and rendering of sound scenes using Ambisonics en- and decoding,
- extraction of sound-field parameters (direction of arrival, diffuseness) from microphone array recordings,
- parametric audio rendering such as the spatial decomposition method, and

[2] `https://jupyter.org/`

- room acoustic simulation using image sources and feedback delay networks.

However, we also welcome and encourage submissions from other areas of acoustics including but not limited to audio signal processing, basics of (technical) acoustics, and the measurement of acoustic properties (transfer functions of linear & time-invariant systems, room acoustic parameters, etc.). Many assignments actively use pyfar functionality, but this is not a strict requirement. For assignments focusing on more basic signal processing, such as introducing the Fourier transform or finding the onset of an impulse response, it may be more appropriate to use low-level Python, `numpy` [3] and `scipy` [4] functionality, and pyfar could serve as a reference implementation against which students can compare their own solutions. On the other hand, high-level assignments that aim to introduce more complex algorithms or audio signal processing, such as measuring an impulse response, may benefit from using pyfar functionality directly to hide some complexity in favor of a clearer and shorter presentation of the required processing blocks.

2.2 Concept

Figure 1 illustrates the concept and technical realization of the pyfar open educational resources. The notebooks initially contain the assignments and their solutions. These notebooks are stored in a private (non-public) git repository to prevent an uncontrolled circulation of the solutions. This is important for teachers who grade assignments as part of their course work. Lecturers will be granted access to the private repository, and thereby the solutions, after having proven that they are teaching a course in acoustics or signal processing. The same holds for teachers who want to contribute an assignment (cf. Section 2.3 for details). Solution-free assignments are made available on the open educational resources section at pyfar.org. They can be accessed by students and interested teachers. The `nbgrader` [5] package is utilized to remove the solutions from the notebooks hosted in the private repository before they are published.

2.3 Requirements

Before an assignment is added to the pyfar open educational resources, it is peer reviewed within the pyfar community to ensure that it meets our imposed requirements. First and foremost, the assignment and its solution should be correct, clear, self-contained, and provide at least one scientifically qualified reference for further reading. Apart from that, a few formalities need to be followed to make the assignments more accessible, such as adding author's contact details, stating the required skills, and providing an estimated time in which the assignment shall be completed. Importantly, the notebook must use `nbgrader` to ensure that the solution can be removed automatically. As an additional benefit, `nbgrader` can be used to automatically grade the students' solution, which could further reduce the lecturer's workload. However, this requires well-defined problems suitable for automatic grading and code blocks that test the solutions. Such tests typically require more design effort. Therefore, we leave the use of this functionality up to the lecturer, as it may not be appropriate for all types of assignments. A sample assignment is

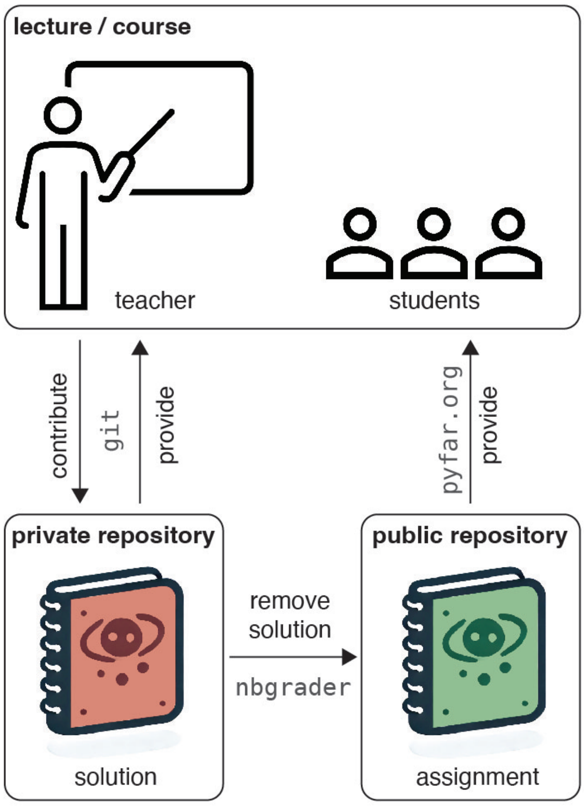

Figure 1. Proposed workflow for providing course materials using the pyfar open educational resources.

[3] `https://numpy.org/`
[4] `https://scipy.org/`
[5] `https://nbgrader.readthedocs.io`

provided to help contributing teachers follow these rules.

2.4 Example

Figure 2 shows a short excerpt from the basic *pyfar challenge* assignment. Like all assignments, it begins with a brief introduction and lists the requirements, the estimated time to complete the task, and the required dependencies. The positions where code has to be added are marked with the comment `#YOUR CODE HERE` and an error will be raised if the block is executed without adding any code. These blocks are automatically generated using `nbgrader` after deleting the solutions.

This specific example might work best as an ungraded assignment at the beginning of a course to familiarize students with Jupyter notebooks, Python, and pyfar. Other assignments could be used in different ways: An assignment could be shown to the class by the lecturer and solved collaboratively during the course. It could also be distributed within a dedicated tutorial to be solved by (groups of) students; or it could be part of the graded course work.

3. CONCLUSION

We introduced the pyfar open educational resources containing programming assignments for teaching (virtual) acoustics and audio signal processing. The assignments are collaboratively developed, maintained, and peer reviewed across different institutes and countries. This may help to improve the quality of teaching and simultaneously reduce the workload of lecturers. The assignments are freely available from `pyfar.org`. At the time of writing, assignments mostly covered topics in virtual acoustics, but we warmly welcome contributions from all other areas of acoustics.

4. REFERENCES

[1] S. Möller, J. Ahrens, E. Altinsoy, J. Fels, G. Müller, G. Reimers, B. Seeber, M. Vorländer, and S. Weinzier, "Konzeption eines MOOC der TU9 zum Thema Communication Acoustics," in *Fortschritte der Akustik – DAGA 2016*, (Aachen, Germany), pp. 527–529, Mar. 2016.

[2] T9, "Communication Acoustics (edX)." `https://www.mooc-list.com/course/communication-acoustics-edx`, 2016. (accessed 2025-03-24).

[3] Acoucou and KFB Acoustics, "Acoucou. Acoustic Courseware." `http://acoucou.com`, 2025. (accessed 2025-03-24).

[4] S. Spors, F. Schultz, V. Erbes, and M. Schröder, "Concept and Realization of Open Educational Resources using Jupyter," in *Fortschritte der Akustik – DAGA 2020*, (Hannover (Remote)), pp. 531–534, March 2020.

[5] J. Biggs and C. Tang, *Teaching for Quality Learning at University*. New York, NY, USA: McGraw-Hill, 4 ed., 2011.

pyfar challenge

The pyfar developers pyfar.org Contact: info@pyfar.org

This assignment will familiarize you with some of the basic pyfar functionality. For help visit the pyfar documentation

Audience: People, who are new to pyfar.

Requirements: No previous knowledge required. Minimal numpy skills are advantageous.

Duration: 45 Minutes

Dependencies `pip install pyfar>=0.7 sounddevice ipympl ipykernel`

```
[ ]: import pyfar as pf
     import numpy as np
     import sounddevice as sd
     import matplotlib.pyplot as plt
     %matplotlib ipympl
```

1. pyfar Signals

pyfar Signal objects store audio data that can be converted between the time and frequency domain. They also contain useful meta data such as the sampling rate or the frequencies at which the spectrum is available. See their documentation for more information

Generate a Signal with a sampling rate of 48 kHz and the time data `[0, 2, 0, 0]`.

```
[ ]: # YOUR CODE HERE
     raise NotImplementedError
```

Inspect your Signal by printing the sampling rate, time data, the times at which the data is available, and the number of samples.

```
[ ]: # YOUR CODE HERE
     raise NotImplementedError
```

Figure 2. Example for an assignment in the form of a Jupyter notebook.