Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

UNIVERSITÄTSKOLLEG

Erfahrungsbericht zum Lehrlabor-Projekt:

JUNOSOL - JUPYTER-NOTEBOOKS FOR SELF-ORGANIZED LEARNING

Projektverantwortliche und Autorinnen und Autoren:

Céline Hadziiouannou (Projektverantwortliche)
celine.hadziioannou@uni-hamburg.de

Johann Jacobsohn
johan.jacobsohn@uni-hamburg.de

Julián Peláez-Quiñones
judape93@gmx.de

Dirk Becker (Erstautor)
dirk.becker-2@uni-hamburg.de

Fakultät für Mathematik, Informatik und Naturwissenschaften

Mai 2020

Das Universitätskolleg wird aus Mitteln des BMBF unter
dem Förderkennzeichen 01PL17033 gefördert.
Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei
den Herausgebern und Autorinnen und Autoren.

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

ABSTRACT

The aims of the project were to introduce the students to a self-organized computational learning environment that they can use for their further (self)studies and to introduce them to the scientific peer review process. Both are skills highly needed in the scientific and professional environments. The self-organized learning was realized with Jupyter notebooks hosted on the DL.MIN hub (https://code.min.uni-hamburg.de/hub/login) and the peer review process and the access to and exchange of all necessary course material was realized with an OpenOLAT room hosted by Universität Hamburg. Jupyter notebooks allow the user to combine data analysis, plotting and interpretation in an easy way and allows for easy reproducibility of the results and easy parameter testing. The whole peer review process can be carried out in a Jupyter notebook provided as exercise. We provided students with screencasts produced in cooperation with the Lehrlabor team and DL.MIN that introduced them to Jupyter notebooks, plotting in python and the peer review workflow as implemented in OpenOLAT. Furthermore, guidelines for peer review and best practices of scientific computing, partly newly developed for the project, were uploaded to the OpenOLAT room and available for the students. The screencasts and an OpenOLAT template course are available.

CONCEPTION AND OBJECTIVES

Using the well-established master module ‘Raum- und Oberflächenwellen-Seismologie (Body and surface wave seismology)’ which is taught in the first year of the Geophysics master, we wanted to introduce the students to the use of Jupyter notebooks and the process of a formalized peer feedback including a grading of the feedback process. The course consists of lectures, compulsory exercises and is normally finalized with a written exam. To be allowed to the final exam, a certain portion of the points in the exercises has to be obtained. As a special motivation, it is possible to obtain bonus points for the written exam by completing the exercises. The exercises are carried out using Jupyter notebooks. The handed in notebooks are swapped among the students and peer reviewed. The final grade of the exercises consists of the grading for the notebooks and the grading given for the peer feedback.

The use of Jupyter notebooks fits nicely into the guiding principle of research-oriented learning championed by Universität Hamburg and CEN (Centrum für Erdsystemforschung und Nachhaltigkeit) of which the Institute of Geophysics is a member. It allows students to acquire skills in an open-source environment that structures the whole process from data acquisition, data processing and plotting up to the final interpretation of their own results. In addition, Jupyter notebooks allow the visualization of scientific results[1] can be used as manuals for computer programs and day-to-day data processing routines[2] and might even be used to reproduce scientific papers or parts of them. In the field of seismology, in which context the funded project is taught, a considerable amount of Jupyter notebooks exits that can be used for self-study.[3] The python programming language used in the Jupyter notebooks in the project is highly popular

[1] LIGO gravitational wave detection in a Jupyter notebook: https://www.gw-openscience.org/LVT151012data/LOSC_Event_tutorial_LVT151012.html

[2] Perkel, J.M. (2018). Why Jupyter is data scientists’ computational notebook of choice. Nature, 563(7729)

[3] L. Krischer et al. (2018). seismo-live: An educational Online Library of Jupyter Notebooks for seismology, Seismol. Res. Lett., doi:10.1785/0220180167

in many scientific disciplines (e.g. obspy[4] in seismology, iris[5] in meteorology) and the fast expanding field of machine learning (e.g. pytorch[6]). Thus, a basic knowledge of Jupyter notebooks and python gives students the opportunity to pursue their own scientific studies and data analysis in a well-organized way. Most of the students participating in the master course already have a basic knowledge in python from the Bachelor course in seismology and some general programming experience from the modules 'Datenverarbeitung und Programmierung in den Geowissenschaften' and 'Numerische Methoden in den Geowissenschaften' which are taught in the 3rd and 4th semester of the Geophysics and Oceanography Bachelor, respectively. The introduction to Jupyter notebooks at the beginning of their Master study might help the students in the self-study of further scientific topics and also facilitate the data handling during a subsequent Master thesis. In the past, already many students in the seismology group employed Jupyter notebooks during their respective Master studies to document their results and organize their data processing. Because Jupyter notebooks also allow for the use of the statistical R software, the self-learning scripts developed during the corresponding Lehrlabor projects 'RLab' und 'RLab 2.0'[7] can also be used to supplement the results obtained using python programs. There are also Jupyter notebooks from the 'RLab' project available for self-study.

Because Jupyter notebooks allow for a system-independent reproduction of data processing and the corresponding visualization of the results when executed on a server, we intended to let the students swap their exercises written in Jupyter notebooks and let them run on a server for peer review. Their review comments can then be directly entered into the Jupyter notebooks and handed back to the supervisors. In case no server solution would be available, we planned to provide a local installation for the students in the terminal room of the Institute of Geophysics or provide a script for a local installation on the students' personal devices.

The assessment of the validity of presented scientific results is an integral part of the scientific process. The peer review introduced within the framework of this project builds upon the modules 'Wissenschaftliches Arbeiten' in the 4th semester and 'Grundlagenseminar' in the 5th semester taught in the Bachelor studies of Geophysics and Oceanography. While these modules introduce some feedback basics, the peer-grading component introduced in this project introduces a more formal approach. This is intended to strengthen a critical inspection of presented results in the students. This builds upon experiences in the former Lehrlabor project 'Peer Grading' by Ulrike von Luxburg[8] who introduced the concept at the introductory level. Results indicated that such an approach endows the students with critical skills like a better self-assessment, critical reading and an ability to swap the perspective in the teacher-student relationship. Here, we introduced the additional approach to 'grade the grading'. The intention was to motivate the students to do thorough feedback work, as it was relevant for their own grades. By providing the students with sample solutions for the exercises in form of Jupyter notebooks (Fig. 1) and letting them do feedback on the same problems they did themselves, we hoped to create a situation in which the students 'as experts' are able to give reasonable feedback. However, it was not intended that this feedback should be

[4] https://github.com/obspy/obspy/wiki

[5] https://github.com/SciTools/iris

[6] https://pytorch.org/

[7] https://rlab.blogs.uni-hamburg.de/

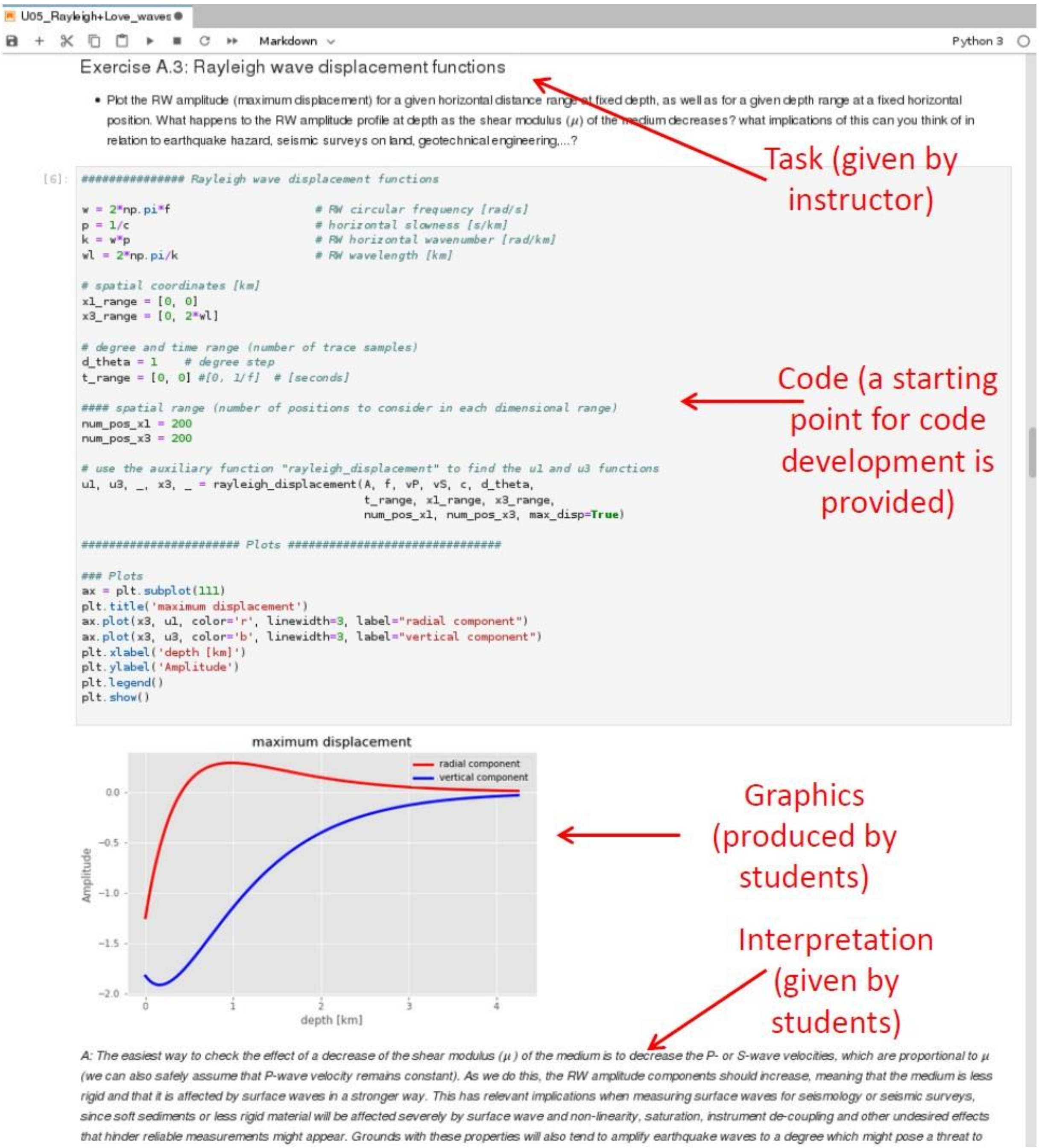
[8] M.S.M. Sajjadi, M. Alamgir, and U. von Luxburg (2016). Peer Grading in a Course on Algorithms and Data Structures: Machine Learning Algorithms do not Improve over Simple Baselines. In Proceedings of the 3rd ACM Conference on Learning @ Scale. pp.369-378, Eds. J. Haywood, V. van Aleven, J. Kay, and I. Roll, Edinburgh, UK.

used to improve the original submission or that the grading given by the peers should be incorporated into the final grade of the student exercises. This grading was still completely up to the course assistant and consisted of the grades for the original submission of the exercise and the performed peer feedback.

The conception for the project included a digital environment in which the students can access all necessary material and software for the solution of the exercises using Jupyter notebooks and the subsequent peer review process. This includes both the Jupyter servers on which the Jupyter software can be executed and some kind of content management system that is able to perform the task of the review process.

To introduce the students to Jupyter notebooks and to ease the peer review process and the file handling during the course, the creation of screencasts (Fig. 2), sample Jupyter notebooks and best-practice handouts for the peer review process (Fig. 3) was intended. The whole conception and implementation of the project was to be carried out in close cooperation of the project group consisting of Prof. Céline Hadziioannou as the project proponent and the responsible lecturer, Johann Jacobsohn as Research Assistant for the technical implementation of the project and the production of the screencasts, Julián Peláez-Quiñones as Student Assistant responsible for the creation of exercise and solution Jupyter notebooks, Fabian Dethof as tutor for the exercises and Dirk Becker for the creation of the course in the content management system.

We planned to provide the students the screencasts and the introductory material for the Jupyter notebooks and the peer review process before the start of the actual lecture period. This should give them the opportunity to get accustomed to the new software and the peer review process by self-study.


U05_Rayleigh+Love_waves
Markdown
Python 3
Task (given by instructor)
Code (a starting point for code development is provided)
Graphics (produced by students)
Interpretation (given by students)
maximum displacement
radial component
vertical component
Amplitude
0.0
-0.5
-1.0
-1.5
-2.0
depth [km]
0
1
2
3
4

Exercise A.3: Rayleigh wave displacement functions

- Plot the RW amplitude (maximum displacement) for a given horizontal distance range at fixed depth, as well as for a given depth range at a fixed horizontal position. What happens to the RW amplitude profile at depth as the shear modulus (μ) of the medium decreases? what implications of this can you think of in relation to earthquake hazard, seismic surveys on land, geotechnical engineering,...?

[6]:

```
############### Rayleigh wave displacement functions

w = 2*np.pi*f                       # RW circular frequency [rad/s]
p = 1/c                             # horizontal slowness [s/km]
k = w*p                             # RW horizontal wavenumber [rad/km]
wl = 2*np.pi/k                      # RW wavelength [km]

# spatial coordinates [km]
x1_range = [0, 0]
x3_range = [0, 2*wl]

# degree and time range (number of trace samples)
d_theta = 1    # degree step
t_range = [0, 0] #[0, 1/f]  # [seconds]

#### spatial range (number of positions to consider in each dimensional range)
num_pos_x1 = 200
num_pos_x3 = 200

# use the auxiliary function "rayleigh_displacement" to find the u1 and u3 functions
u1, u3, _, x3, _ = rayleigh_displacement(A, f, vP, vS, c, d_theta,
                                          t_range, x1_range, x3_range,
                                          num_pos_x1, num_pos_x3, max_disp=True)

####################### Plots ###############################

### Plots
ax = plt.subplot(111)
plt.title('maximum displacement')
ax.plot(x3, u1, color='r', linewidth=3, label="radial component")
ax.plot(x3, u3, color='b', linewidth=3, label="vertical component")
plt.xlabel('depth [km]')
plt.ylabel('Amplitude')
plt.legend()
plt.show()
```

A: The easiest way to check the effect of a decrease of the shear modulus (μ) of the medium is to decrease the P- or S-wave velocities, which are proportional to μ (we can also safely assume that P-wave velocity remains constant). As we do this, the RW amplitude components should increase, meaning that the medium is less rigid and that it is affected by surface waves in a stronger way. This has relevant implications when measuring surface waves for seismology or seismic surveys, since soft sediments or less rigid material will be affected severely by surface wave and non-linearity, saturation, instrument de-coupling and other undesired effects that hinder reliable measurements might appear. Grounds with these properties will also tend to amplify earthquake waves to a degree which might pose a threat to

Fig. 1: Sample solution Jupyter notebook produced for the project. For exercises, students were provided with a Jupyter notebook that detailed the tasks to address as homework, gave some background explanations and additional resources and a starting point for code development. Students completed the code, produced corresponding graphics, interpreted their results and handed in the final notebooks for peer review and review by the student tutor.

IMPLEMENTATION

In close cooperation with the DL.MIN team and the team from the Lehrlabor, we discussed several possible implementations of both the document handling during the peer review process and the implementation of the Jupyter notebook software. However, as there exists no single system that addresses both requirements at the same time, we used a dual setup of OpenOLAT for the course organization and a JupyterHub for the actual programming and review tasks. OpenOLAT is an e-learning platform widely used at the UHH and helped us manage our course by setting up and assigning exercises to each student. It also served to host additional material (screencasts, documents etc.) and tracking assignment progress. Because this is an already existing system, which is widely used, we preferred it with respect to other solutions like an implementation with GitHub that was also in the discussion. We also set up a trial routine under GitHub and tested it internally in our group. However, we finally decided against such an implementation, as it would have required a local installation on a student's device and a probably steeper learning curve than the OpenOLAT implementation.

To introduce the students to OpenOLAT and the handling of the peer review process in OpenOLAT, we produced a screencast walkthrough. This screencast was given to the students as homework. In addition, we also invested the first in-person exercise lesson to introduce the students to OpenOLAT and the downloading and uploading of material. We created a sample exercise (Exercise 0) that we used to acquaint the students with the technical procedure of the peer review process. One of the reasons for this special emphasis on teaching the technical implementation was that OpenOLAT does not provide a native module or functionality for a peer feedback process. Our workaround was to use the workflow of the exercise module (Kursbaustein Aufgabe) in OpenOLAT and to formally refuse the initial submission of the students. This sends the task back to the students for revision. Fig. 4 shows the workflow implemented in our OpenOLAT course. The task was initially uploaded as a Jupyter notebook to the OpenOLAT course with a fixed date for handing in the exercise ('Zuweisung Aufgabenstellung'). Students downloaded the Jupyter notebook created by the student assistant containing the tasks, additional information and a starting point for code development (see Fig. 1). They uploaded the finished Jupyter notebook ('Abgabe'). The notebooks were swapped among the students automatically by a script written by Johann Jocobsohn,[9] which allows for an automatization of this task by downloading all student notebooks at once and performing an allocation that avoids an assignment of their own submission. The script even performs an initial check of the functionality of the submitted notebooks and contacts students with faulty submissions automatically by mail. The review files produced by this process are manually uploaded again to the OpenOLAT course room (file BAR6178.zip under 'Rückgabe und Feedback' in Fig. 4). Students review their peers' submission and upload it again to OpenOLAT (file Review4.zip under 'Überarbeitung'). Finally, the student tutor reviews both the original submission (U04_BAT1234.zip in Fig. 4) and the peer review (Review4.zip) and uploads his corresponding comments as html-files exported from the Jupyter notebook server (files BAR6178_reviewed_*.html and u04_*_corrected.html under 'Überarbeitung' in Fig. 4). Now the submission is formally accepted, and a grading entered in OpenOLAT ('Bewertung'). During the revision stage, which starts after the deadline for handing in the exercise, students have access to a sample solution provided online in OpenOLAT ('Musterlösung'). The sample solution also consisted of a Jupyter notebook prepared by the student assistant.

JupyterHub is a hosted version of Jupyter notebooks, which is the environment in which programming tasks are solved by our students. JupyterHub helped us provide a working setup to all students with minimal maintenance. We used the DL.MIN provided JupyterHub[10] that is also able to run Jupyter R scripts and

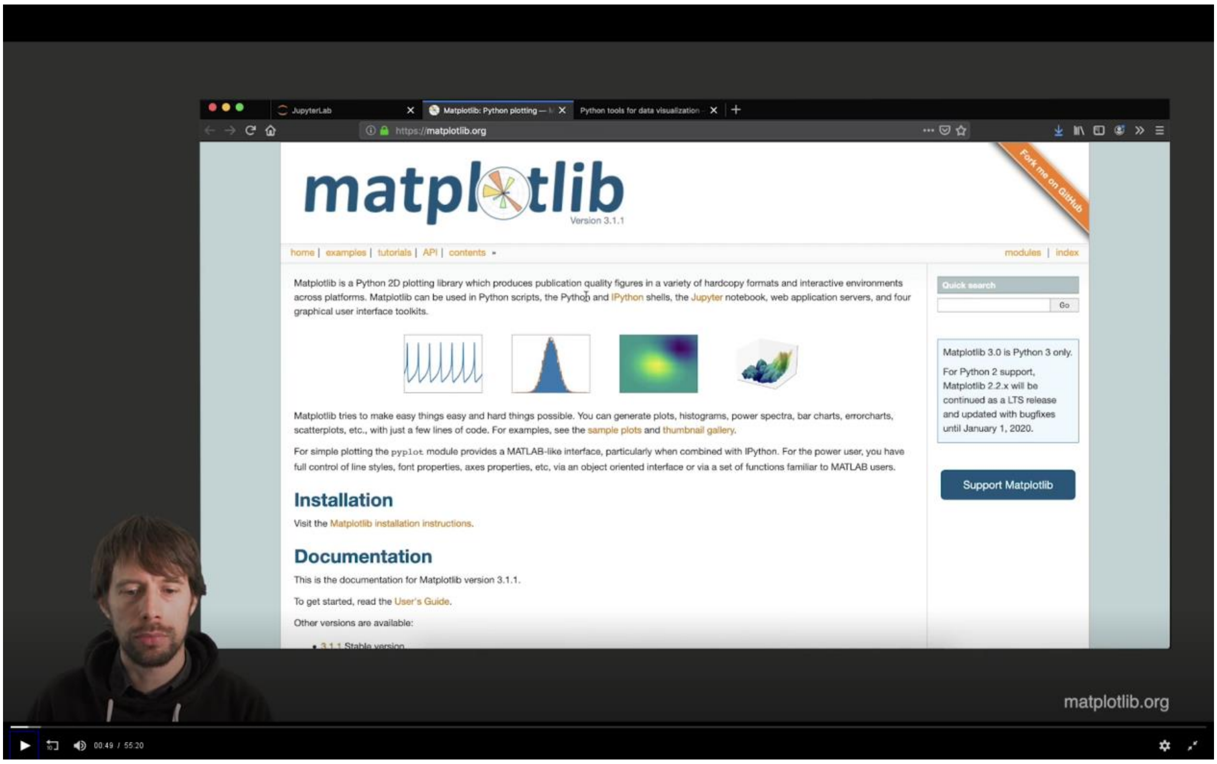
[9] https://gitlab.rrz.uni-hamburg.de/jjacobsohn/assign_reviews

[10] https://code.min.uni-hamburg.de/hub/login

was already tested in the RLab Lehrlabor projects. Due to the great support from the DL.MIN team (Michael Heinicke and Lars Thoms) a python environment was up and running on the JupyterHub at the start of the semester that included all necessary python modules to run the Jupyter notebook exercises. The JupyterHub is also able to incorporate further python modules when needed. We introduced the students to the functionalities (login, file upload, modifying and running Jupyter notebooks) of the JupyterHub during the first in-person exercise session and provided them sample Jupyter notebooks for testing of the environment. During the course, students uploaded the Jupyter notebooks provided in the OpenOLAT course room to the JupyterHub, worked on them online and downloaded the JupyterNotebook file for later submission to the OpenOLAT system. There is also the option to export PDF or HTML versions of the Jupyter notebook. Because the JupyterHub is a server implementation, no local installation by the students is needed. Thus, both the data handling in the OpenOLAT course room and the work on the Jupyter notebooks on the JupyterHub can be performed on a device with internet access in a normal browser window.

Before the start of the lecture period, supporting screencasts and handouts were created with the intention to help the students with running Jupyter notebooks, understanding the handling of the peer review process in OpenOLAT and giving them general guidelines for the peer review process. Johann Jacobsohn produced Screencasts in close cooperation with the DL.MIN team (Karen Kandzia, Stefan Koch, Michael Heinicke) and Christian Kreitschmann from the Lehrlabor team. These screencasts were available as download from the OpenOLAT course room and students were asked to review these screencasts before starting their work on the exercises. These screencasts were available throughout the course for reference. In addition, a compilation of useful references for python, plotting in python and Jupyter notebooks was provided along with a document containing coding tips provided as both pdf and executable Jupyter notebook. These documents were also made available in a resources folder in the OpenOLAT room for self-study. Finding common ground on the question what characterizes a good review took several iterations within the project team. Especially the question how a proper review for computer code should be performed proved difficult. We finally ended up with creating handouts for students that give guidelines for the scientific part, for the implementation of the code and the graphical presentation of the results (Fig. 3) as well as a document that contains some general ideas about the review of computer code (available in the template OpenOLAT course room). In addition, we also provided some general best practice recommendations for peer review and profited from discussions with Matthias Otto and several documents he produced for projects of the Universitätskolleg[11] that we also linked for the students. We also provided several documents with more course specific resources dealing with the scientific content of the lecture (body and surface waves). The students could also use these for self study.

[11] https://sumo.blogs.uni-hamburg.de/?p=2500



Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Installation

Visit the Matplotlib installation instructions.

Documentation

This is the documentation for Matplotlib version 3.1.1.

To get started, read the User's Guide.

Other versions are available:

- 3.1.1 Stable version

Matplotlib 3.0 is Python 3 only.

For Python 2 support, Matplotlib 2.2.x will be continued as a LTS release and updated with bugfixes until January 1, 2020.

Fig. 2: Screenshot from screencast giving an introduction to plotting with python.

RESULTS AND DELIVERABLES

During the course of the project Johann Jocobsohn produced several screencasts (Fig. 2), which detail a) the process of obtaining an assignment from OpenOLAT, working on this assignment on the DL.MIN JupyterHub and uploading the solutions back to OpenOLAT, b) doing the review process in the course, c) a basic introduction to Jupyter notebooks and d) some general information about plotting in python. The former two screencasts might be used in other courses to introduce students to OpenOLAT, Jupyter notebooks and a peer review process modelled to our project. The other two notebooks can be used for a general introduction to the use of python in Jupyter notebooks. Spending the first exercise lesson and the first (not graded) exercise in general to introduce the students to the workflow of the peer review proved beneficial. Because of the formal rejection of the initial student submission, this workflow was not entirely intuitive. A specialized OpenOLAT module would be helpful here. The production of the screencasts consumed quite some time and things like the different pitch of the speaker during different times of the day have to be considered when producing longer screencasts. One of the lessons learned from the student feedback is a desire for shorter screencasts that only address a specific topic and not extended ones that deal with several at once. However, a structuring of specific contents of the screencasts might be done via chapters that can be referenced in the OpenOLAT course.

Providing the students with Jupyter notebooks that contained the exercises and a starting point for the code development was also good. It shifted the focus a little bit from the formal set up of the notebook environment (loading basic modules, setting up the correct parameters for the display of the graphics) to working on the scientific code implementation.

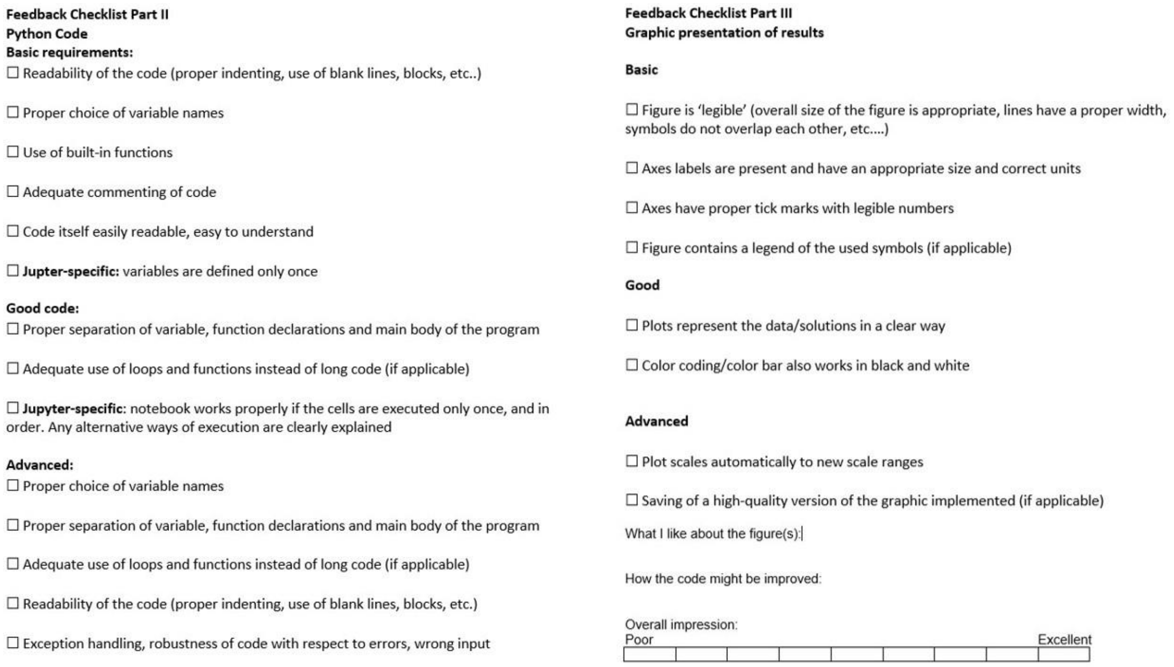
Students also generally gave good and constructive peer feedback and the given grading was comparable to that given by the student tutor. However, the first few exercises were not so easy to review, as there was not so much scientific interpretation involved. This emphasizes the need to come up with proper exercises for review.

The students in general were content with the presented concept of self-organized learning with Jupyter notebooks and the peer feedback process. They had only minor problems with the workflow and were generally able to submit their exercises and reviews without any problems. However, an inclusion of the given peer feedback into a revision of the exercise before final submission might increase the students' motivation to deal with the given feedback. Due to the small number of students participating in the first round of this new course layout, it is not possible to tell whether the new peer feedback concept that also had the intention to better familiarize the students with the lecture material by inspecting their peers' solution while being aware of the master solution, really worked. However, the general feedback was positive and possible positive effects might be investigated during the coming years when the material produced during the Lehrlabor project phase will be further used and possibly modified.

Participants were also polled informally to fine-tune the process and were encouraged to contact staff personally or via email for further explanation, where needed. This was possible because there was a close contact between the students and the lecturer and the student tutor during the lectures and exercises due to the small number of students participating in the exercises. This might be more difficult in a larger course with more students.

While the students worked on and completed nearly all the exercises, they did not follow up on many of the self-learning materials also presented in the course environment. They mostly stuck with the material needed to complete the exercises. This might be due to high workload for students during their Master studies. Thus, it might be helpful to limit the additional resources to a few selected examples. However, several students incorporated the use of Jupyter notebooks for their own studies and then also used additional material for self-study as intended.

Screencasts, guidelines for peer review, additional material residing in a resources OpenOLAT course room resources folder and a short introduction for the peer review workflow were copied into a template OpenOLAT room for later use during future semesters of this course and also possibly for other courses. This template course room can also be accessed.



Feedback Checklist Part II
Python Code
Basic requirements:

☐ Readability of the code (proper indenting, use of blank lines, blocks, etc..)

☐ Proper choice of variable names

☐ Use of built-in functions

☐ Adequate commenting of code

☐ Code itself easily readable, easy to understand

☐ **Jupter-specific:** variables are defined only once

Good code:

☐ Proper separation of variable, function declarations and main body of the program

☐ Adequate use of loops and functions instead of long code (if applicable)

☐ **Jupyter-specific**: notebook works properly if the cells are executed only once, and in order. Any alternative ways of execution are clearly explained

Advanced:

☐ Proper choice of variable names

☐ Proper separation of variable, function declarations and main body of the program

☐ Adequate use of loops and functions instead of long code (if applicable)

☐ Readability of the code (proper indenting, use of blank lines, blocks, etc.)

☐ Exception handling, robustness of code with respect to errors, wrong input

Feedback Checklist Part III
Graphic presentation of results

Basic

☐ Figure is 'legible' (overall size of the figure is appropriate, lines have a proper width, symbols do not overlap each other, etc....)

☐ Axes labels are present and have an appropriate size and correct units

☐ Axes have proper tick marks with legible numbers

☐ Figure contains a legend of the used symbols (if applicable)

Good

☐ Plots represent the data/solutions in a clear way

☐ Color coding/color bar also works in black and white

Advanced

☐ Plot scales automatically to new scale ranges

☐ Saving of a high-quality version of the graphic implemented (if applicable)

What I like about the figure(s):

How the code might be improved:

Overall impression:
Poor Excellent
☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

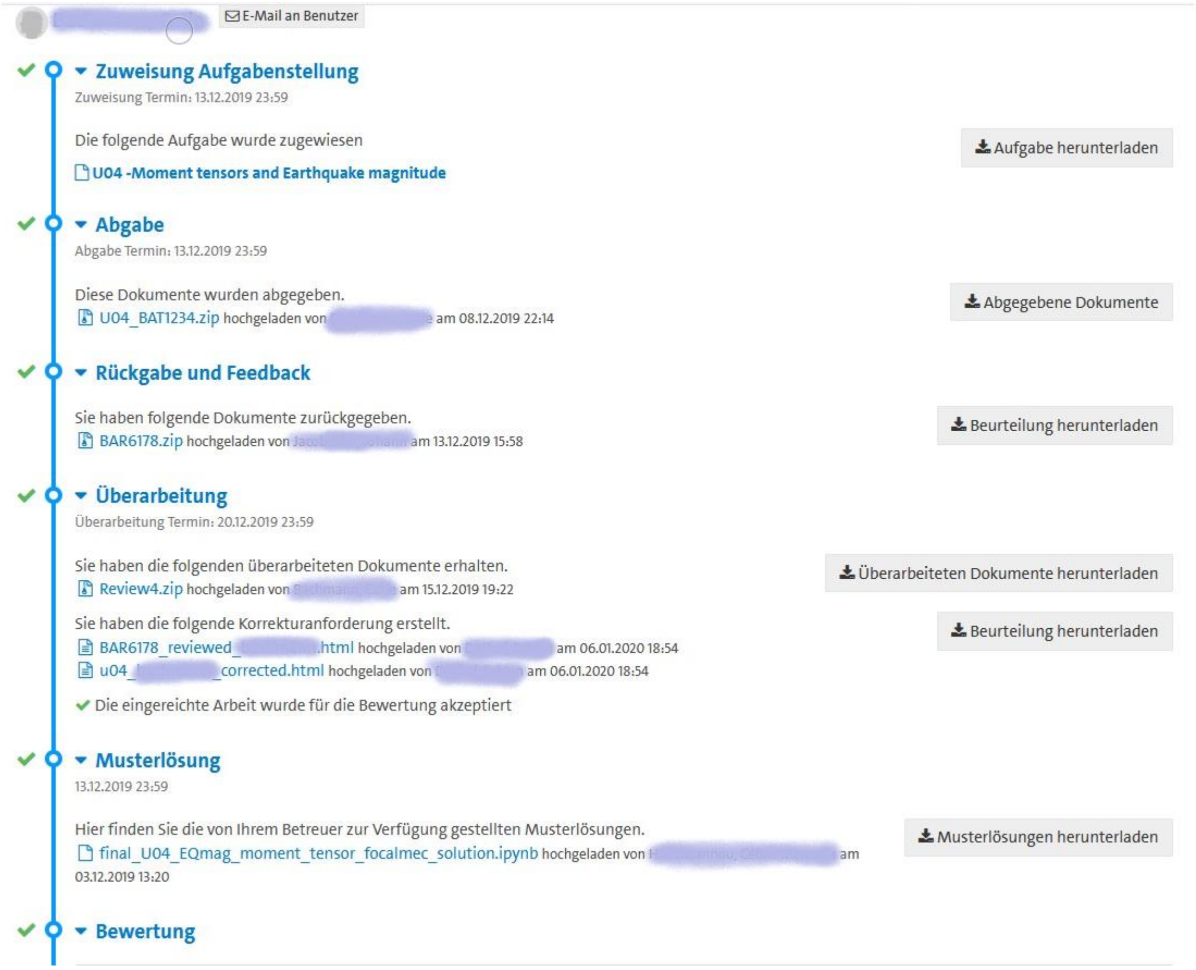
Fig. 3: Handouts for peer review guidelines for the code implementation (left) and graphic representation of the results (right) in the Jupyter notebooks.

RETROSPECTION AND OUTLOOK

In general, students were quick to adopt Jupyter notebooks into their daily work and in the subsequent semester, several students also presented their seminar work in Jupyter notebooks. Thus, the introduction of Jupyter notebooks for self-organized study and documentation of their work can be regarded as (at least partly) successful.

The choice of the dual set up with OpenOLAT for the organization of the course material and the review process and the JupyterHub of DL.MIN for running the Jupyter notebooks proved beneficial. Although a single system would be preferable (like running a JupyterHub in OpenOLAT or Moodle), the students had no problem to work with the two different systems and encountered only minor problems during the first sessions. Both systems worked fine with only minor exceptions. We once managed to break the JupyterHub by processing an excessively large dataset. However, the DL.MIN team quickly rectified this. Due to the move to Moodle as a course management system within the Institute of Geophysics during the introduction of digital teaching during the Corona lockdown, there might be a shift to Moodle in the future. An implementation in Moodle might benefit from the 'Gegenseitige Beurteilung' activity implemented there, although we did not yet have time to test the feasibility of such an implementation.

It is intended to perpetuate the approach developed during the Lehrlabor project phase by teaching the master module in future semesters in the same way. The choice of exercises might be improved to give the students the opportunity for a more meaningful feedback. This would imply a stronger focus on the interpretation of the results and not only the technical aspects of algorithm implementation and the graphical representation of the results. This was also one of the critical remarks raised by the students. The exercises might thus be modified to give the students a better opportunity to judge the scientific results and interpretation of their peers and not only the implementation of the code. However, as the introduction to the use of Jupyter notebooks and good programming style will still be a main objective in coming years, we will keep some emphasis on proper coding which is also vital during the following Master studies.


E-Mail an Benutzer
Aufgabe herunterladen
Abgegebene Dokumente
Beurteilung herunterladen
Überarbeiteten Dokumente herunterladen
Beurteilung herunterladen
Musterlösungen herunterladen

[redacted]

✓ ▾ Zuweisung Aufgabenstellung
Zuweisung Termin: 13.12.2019 23:59

Die folgende Aufgabe wurde zugewiesen
U04 -Moment tensors and Earthquake magnitude

✓ ▾ Abgabe
Abgabe Termin: 13.12.2019 23:59

Diese Dokumente wurden abgegeben.
U04_BAT1234.zip hochgeladen von [redacted] am 08.12.2019 22:14

✓ ▾ Rückgabe und Feedback

Sie haben folgende Dokumente zurückgegeben.
BAR6178.zip hochgeladen von [redacted] am 13.12.2019 15:58

✓ ▾ Überarbeitung
Überarbeitung Termin: 20.12.2019 23:59

Sie haben die folgenden überarbeiteten Dokumente erhalten.
Review4.zip hochgeladen von [redacted] am 15.12.2019 19:22

Sie haben die folgende Korrekturanforderung erstellt.
BAR6178_reviewed_[redacted].html hochgeladen von [redacted] am 06.01.2020 18:54
u04_[redacted]_corrected.html hochgeladen von [redacted] am 06.01.2020 18:54

✓ Die eingereichte Arbeit wurde für die Bewertung akzeptiert

✓ ▾ Musterlösung
13.12.2019 23:59

Hier finden Sie die von Ihrem Betreuer zur Verfügung gestellten Musterlösungen.
final_U04_EQmag_moment_tensor_focalmec_solution.ipynb hochgeladen von [redacted] am 03.12.2019 13:20

✓ ▾ Bewertung

Fig. 4: Workflow of the peer review process as implemented in the exercise module (Kursbaustein Aufgabe) in OpenOLAT.

There might also be a revision of the peer review process by giving the students the opportunity to incorporate the obtained feedback into their final submission. By only grading the original submission and the given feedback, the motivation to access the feedback for improving one's own understanding of the subject might be somehow diminished. This might shift a bit the focus from the grading and evaluation of the feedback itself and will focus more on the whole process of revising an exercise based on the given feedback and might even improve the students' understanding of the studied subject. There are experiences in our institute with feedback schemes during the preparation of Master theses where students are also graded for the quality of their feedback.

It would be nice to check the technical implementation of the automatized reviewer assignment in a larger course. This will also allow for a truly anonymous review, which was not possible in the small group of our course. However, as the technical implementation now exists, everyone is free to download it from the university GitLab page and to try it.

ACKNOWLEDGEMENTS

We would like to thank the whole Lehrlabor team for the great support during the complete project phase from the initial submission up to the creation of the final report. We are especially indebted to Christian Kreitschmann for many fruitful discussions concerning the practical implementation of the project and the creation of screencasts and other course material. We also want to thank Manuela Kenter and Carolin Gaigl for their support in the organization of the project, many useful discussions, and the facilitation of helpful contacts. The DL.MIN team provided the JupyterHub used for running the Jupyter notebooks and provided the hardware and invaluable assistance in the creation of the screencasts. Michael Heinecke and Lars Thoms took great care of all our requests to create a JupyterHub custom-tailored to our specific requirements and even fixed the server in their free time when we broke it by our own fault. Karen Kandzia and Stefan Koch assisted in the production of the screencasts and introduced Johann to the technical equipment available. Matthias Otto from the Universitätskolleg provided valuable input for the peer review process implemented in the project. Last but not least, we want to thank the students of the Geophysics Master course '63-650 Body and surface wave seismology' in the winter semester 2019/20 for their enthusiasm and willingness to take part in the first round of this Jupyter-notebook-peer-review endeavor. Lessons learned from and feedback given by them will improve the course in the future.