Self-Contained Jupyter Notebook Labs Promote Scalable Signal Processing Education

Dominic Carrano, Ilya Chugunov, Jonathan Lee, Babak Ayazifar
Department of Electrical Engineering and Computer Sciences (EECS), University of California, Berkeley, USA.

Abstract

Our upper-division course in Signals and Systems at UC Berkeley comprises primarily second and third year undergraduates and assumes only a basic background in electrical engineering and computer science. We have introduced Jupyter Notebook Python labs to complement the more theoretical material covered in lectures and homeworks.

Courses at other institutions have created labs with a similar goal in mind. However, many have a hardware component or involve in-person lab sections that require teaching staff to monitor progress. This presents a significant barrier for deployment in larger courses. Virtual labs—namely pure-software assignments and the Jupyter Notebook framework—recently emerged as a solution to this problem. Some courses use programming-only labs that lack the modularity and rich user interface of Jupyter Notebook's cell-based design. Other labs based on Jupyter Notebook have not yet tapped the full potential of its versatile features.

Our labs (1) demonstrate real-life applications; (2) cultivate computational literacy; and (3) are structured to be self-contained. These design principles reduce overhead for teaching staff and give students relevant experience for research and industry.

Keywords: *Python, Jupyter, virtual lab, educational technology, electrical engineering, computer science*

1. Introduction

1.1. Signals and Systems at UC Berkeley

Undergraduates in the Electrical Engineering and Computer Sciences (EECS) Department at UC Berkeley must take six lower-division courses: (1, 2) Designing Information Devices and Systems I and II; (3) Discrete Mathematics and Probability Theory; (4) Structure and Interpretation of Computer Programs; (5) Data Structures; and (6) Machine Structures. The first three courses constitute a design and modeling trilogy and the latter three a computation trilogy. Students take these two sequences in parallel during their first four semesters. Accordingly, our course in *Signals and Systems* assumes only completion of the first two courses in each sequence, constituting a background in linear algebra, basic modeling skills, and two semesters of programming experience. The course covers linear time-invariant (LTI) system theory, Fourier analysis, Shannon-Nyquist sampling theory, and system analysis using the Z and Laplace transforms, and serves as a gateway to the upper-division EECS curriculum, including more advanced courses such as *Digital Signal Processing* and *Feedback Control Systems*.

In Fall 2015, the Department's lower-division EE curriculum was restructured, with the courses *Structure and Interpretation of Signals and Systems* and *Introduction to Microelectronic Circuits* replaced by *Designing Information Devices and Systems I and II*. Previously, the Department offered two signals and systems courses, one in the lower-division and one in the upper-division. The lower-division course had a significant lab component (Lee & Varaiya, 2003; Liu *et al.*, 2010) with weekly sections where students completed in-person MATLAB or LabView assignments. Many other institutions use something similar in their counterpart courses. After the 2015 revision to the curriculum, students in our upper-division course in Signals and Systems continued to engage with the material primarily through written homework assignments and weekly discussion sections. However, until recently, the course continued without a substantive lab component.

1.2. Design Considerations

Around the time our Department restructured its lower-division curriculum, Jupyter Notebook (Kluyver *et al*., 2016) was published as a spin-off of the IPython suite (Pérez & Granger, 2007), providing an interactive platform for scientific computing ("Project Jupyter", 2020). Since its initial release, Jupyter Notebook has exploded as an education tool in fields such as signal processing, machine learning, and artificial intelligence (Lovejoy & Wickert, 2015; O'Hara *et al*., 2015; Granado *et al*., 2018; Herta *et al*., 2019). Even within our EECS Department, several other courses, such as *Probability and Random Processes* and *Digital Signal Processing*, use assignments based on Jupyter Notebook.

After considering our course's role in the EECS curriculum, we specified three requirements for using Jupyter Notebook's rich features to reintroduce a hands-on lab component to our course. First, students should connect lab material to real-world applications, tracing the path from abstract mathematical concepts to concrete engineering.

Second, the programming tasks within the labs should foster a proficiency in the modern scientific computing libraries used in industry and research. Just as the physical labs in a circuits course teach students the ins and outs of hardware tools such as oscilloscopes and function generators, virtual labs in our Signals and Systems course should highlight the techniques essential to modern software implementations such as vectorized algorithms.

Third, the labs should be self-contained so they do not demand excessive hand-holding by the teaching staff. This enables scaling to higher enrollments, especially where access limitations to personnel, physical space, and other resources exist.

In this paper, we offer an in-depth tour of one of our labs. We discuss how, through a delicate interplay of Python code, embedded media, and graphical representations of data, we have created virtual labs that take advantage of the strengths of Jupyter Notebook to enrich student learning. The physical and virtual labs that we have surveyed fall short in at least one of these three design principles. Creating virtual labs that satisfy these criteria has allowed us to not only cover the gamut from theory to state-of-the-art applications, but also keep administrative overhead in check.

2 . The Labs

2.1. Related Work

Comparable signals and systems courses have often used lab work to reinforce theoretical content covered in lecture:

- Intro to Signal Processing at Georgia Tech featured MATLAB coding labs that extend conventional pencil-and-paper problem sets ("Lab Assignments for ECE 2026", 2018). Although most labs have a strong application focus and concrete visual or auditory components, they also demand substantive in-person instructor time for verification of multiple checkpoints.
- Signals and Systems at Princeton has several MATLAB-based labs ("ELE 301", 2011), some of which require an in-person checkoff of either a hardware or a software implementation.
- 6.003 at MIT complements problem sets with online Python programming exercises ("Signal Processing", 2019). As students implement most exercises using Python primitives, they are shielded from using Python's open-source scientific computing

libraries, such as NumPy and SciPy for multidimensional arrays and signal processing and Matplotlib for drawing plots.

Jupyter Notebook is a useful tool to teach signal processing concepts. However, many collections of notebooks available online, such as the companion GitHub repository for the book "Python for Signal Processing" (Unpingco, 2014), are not directed primarily at classroom applications. Furthermore, many are not self-contained. For example, some use mathematical knowledge, such as probability, without sufficient background information.

2.2. Summary of Contributions

Table 1 summarizes Jupyter Notebook labs deployed in the fall 2019 semester. Each Notebook is a single document that consists of text, code, and embedded media "cells" that are rendered or executed. Although the teaching staff provides support in office hours and through the Piazza online discussion platform, most students complete the labs autonomously because of the labs' self-contained design:

- Specifications are narrow and well-defined, often on a function-by-function basis, to guide students to implement a sophisticated application without going off-track.
- Hints about useful scientific computing library routines and in-text hyperlinks to library documentation build student fluency with computational tools.
- Numerical test cases and juxtapositioned plots of expected and actual results help students verify their progress without giving away the algorithm, reducing debugging frustration.
- Extensive references allow students to dive into the literature that inspired the labs.

Table 1. Labs for Signals and Systems at UC Berkeley, Fall 2019.

Lab	Topics
Introduction to Python for Signals and Systems	Essentials of the Python programming language and scientific libraries, rectangular/exponential signal generation, convolution
Applications of LTI Filtering	1D edge detector, simple moving average for denoising, exponential moving averaging of stock price data
Fast Fourier Transform (FFT)	Naive Discrete Fourier Transform and FFT implementation, runtime analysis, virtual oscilloscope calibration
Heart Rate Monitoring	Spatial averaging of video of patient's thumb for dimensionality reduction, extracting heartbeat frequency through FFT
(De)convolution	Convolution (image blurring and sharpening), deconvolution (audio echo cancellation, Hubble telescope image deblurring)

Controls	Closed-loop system analysis, signal filtering, root locus analysis, feedback control of a virtual inverted pendulum

Source: EE120 Home Page (2019).

3. Case Study: (De)convolution Lab

Deconvolution and image processing are extensions of the explicit scope of the course. However, we release this lab later in the semester as a wider exploration of the field. The lab augments the students' understanding of the operations and transforms taught in the course by introducing them to higher-dimensional problems and unexpected contexts.

3.1. Real-World Applications

Students often ask: "Can we *undo* convolution?" This lab presents two deconvolution problems to them. The first is to remove echoes from corrupted audio data (Eneroth, 2001). The second extends 1D concepts learned in the first problem to deblur 2D image data. Mirror imperfections of the $1.5B Hubble Space Telescope originally produced blurred deep-space images, as shown in Figure 1(a). Inspired by the solutions proposed at the time by experts in the field (e.g., White, 1992), the lab teaches students how to apply deconvolution algorithms to deblur these images. Figure 1(b) shows a deblurred image.

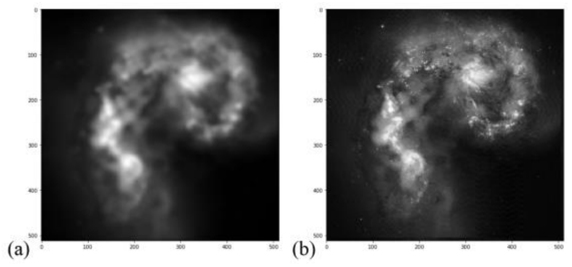

Figure 1. Space image deconvolution results. Source: ESA/Hubble, distributed under CC BY 4.0

Students reproduce these exciting results through a series of building blocks, such as signal quantization, subtractive image sharpening, and Gaussian blurring—each of which is an important signal processing application in its own right.

3.2. Computational Literacy

Before students implement sophisticated algorithms, they learn how to load, and understand the structure of, the underlying signal data. In the audio section, they learn how to use the `scipy.io.wavfile` module to read and write a WAV file, understand its matrix dimensions, and recognize its underlying sampling rate. In the image section, they learn how to use the `plt.imread` function to read an image and understand how its data matrix renders on the screen. Figure 2 illustrates how a 1D integer array is converted to a 2D matrix, and how each matrix entry is mapped to a pixel luminance.

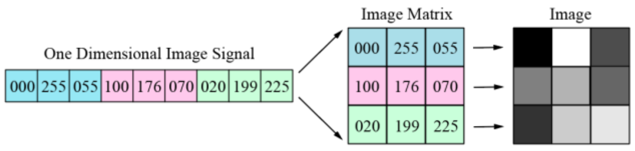

Figure 2. Guided example of understanding underlying data signal for an image. Source: Own elaboration.

Next, the lab shows students how to use, and understand the outputs of, functions from the `np.fft`, `np.linalg`, and `scipy.signal` modules by solving short subproblems. Students demonstrate their comprehension in subsequent larger programming sections. To reinforce the power of these libraries, the final section of the lab cautions students that their code, if implemented incorrectly or inefficiently, may take too long to converge to a correct solution. An efficient solution, which invokes these libraries, takes only seconds.

A drawback of functions in high-level scientific computing libraries that they often accept an unwieldy number of parameter combinations. Through analysis of student submissions and feedback on previous iterations of the lab, we discovered multiple ways to tackle this problem. At the first appearance of each function, so that students can deep-dive into its inner workings, we embed in the lab direct links to relevant documentation. As the libraries are open-source, students can even study the full source code of each function, if they wish.

Sometimes we instruct students to purposefully introduce errors into their code and reflect on the results. In this lab we have students perform Gaussian blurring of an image, Figure 3(a), with convolution parameter ‘full’, rather than the recommended ‘valid’. We ask them to comment on the sudden appearance of black edges in the resultant image, Figure 3(b).

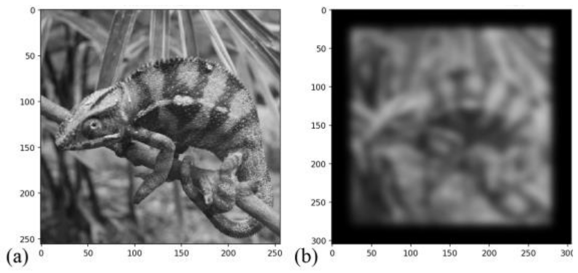

Figure 3. Example of purposeful error. Source: Own elaboration.

This both teaches the effects of parameter modifications, and develops debugging skills when students identify similar errors in later exercises.

Finally, we introduce the parameter sweep, an effective brute force method. Oftentimes there is no mathematically calculable 'best value' for a parameter—as is the case when choosing a Gaussian filter for the final deconvolution problem in this lab—and the student's best option is to sweep a range of values to identify which ones produce acceptable results.

3.3. Self-Containment

Jupyter Notebooks eliminate many traditional scalability problems, such as material distribution and access moderation. However, the number of students who require staff assistance still grows with enrollment, which can bottleneck course expansion.

The lab guides students through tasks ranging from loading audio data to writing a nested parameter sweep, building skills as students need them to complete all the exercises. Furthermore, its cell-based design allows for students to complete some sections of the Jupyter Notebook in a nonsequential order. For example, if they are stuck on the audio section, they can take a break from it and work on the imaging sections.

4. Conclusions and Future Work

We have showcased an approach to designing labs that emphasizes applications and a flavorful use of open-source scientific computing libraries to train students and motivate their learning for research and industry. Crucially, the labs are scalable because they are self-contained; all the resources necessary to complete the lab are embedded within it, and we can deploy them with limited administrative overhead.

We currently have six labs. In the future, we plan to create four additional labs and publish all ten online as freely available resources so our colleagues at other institutions can incorporate them into their courses and provide constructive feedback. We also plan to integrate Jupyter Notebook tools such as the *nbgrader* project (https://nbgrader.readthedocs.io) into our labs, allowing us to automate lab grading in future semesters.

References

EE120 Home Page. (2019). Retrieved January 17, 2020, from http://inst.eecs.berkeley.edu/~ee120/fa19/

ELE 301: Signals and Systems - Labs. (2011). Retrieved January 17, 2020, from https://www.princeton.edu/~cuff/ele301/labs.html

Eneroth, P. (2001). Stereophonic Acoustic Echo Cancellation: Theory and Implementation. Sweden, KFS AB, Lund.

Granado, E. C., & García, E. D. (2018). Guide to Jupyter Notebooks for educational purposes (Á.V. Fernández, Trans.). Retrieved from Universidad Compultense de Madrid Web site: https://eprints.ucm.es/48305/1/ManualJupyterIngles.pdf.

Herta, C. *et al*. (2019). Deep Teaching: Materials for Teaching Machine and Deep Learning. *5th International Conference on Higher Education Advances (HEAd'19),* 1153-1161. doi: 10.4995/HEAd18.2019.9177.

Kluyver, T. *et al*. (2016). Jupyter Notebooks—a publishing format for reproducible computational workflows. *Proceedings of the 20th International Conference on Electronic Publishing.* doi:10.3233/978-1-61499-649-1-87.

Lab Assignments for ECE 2026. (2018). Retrieved January 17, 2020, from https://barry.ece.gatech.edu/2026/labs/

Lee, E. A., & Varaiya, P. (2003). *Laboratory Manual to Accompany "Structure and Interpretation of Signals and Systems."* Boston, MA: Addison Wesley (Pearson).

Liu, H., Kotker, J., & Ayazifar, B. (2010). A First Lab in Filter Design: Power Line Hum Suppression in an ECG Signal. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. doi:10.1109/ISCAS.2010.5536981.

Lovejoy, M. R., & Wickert, M. A. (2015). Using the IPython notebook as the computing platform for signals and systems courses. *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*. doi:10.1109/DSP-SPE.2015.7369568.

O'Hara, K. J., Blank, D., & Marshall, J. (2015). Computational notebooks for AI education. *In Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*.

Pérez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing Science & Engineering*, 9(2), 21-29. doi:10.1109/MCSE.2007.53.

Project Jupyter. (2020, January 23). In *Wikipedia, The Free Encyclopedia*. Retrieved January 24, 2020, from https://en.wikipedia.org/w/index.php?title=Project_Jupyter.

Signal Processing. (2019). Retrieved January 17, 2020, from https://sigproc.mit.edu/fall19

Unpingco, J. (2014). *Python for Signal Processing*. New York, NY: Springer.

White, R. L. (1992). Restoration of images and spectra from the Hubble Space Telescope. In *Astronomical Data Analysis Software and Systems I* (Vol. 25, p. 176).