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 sophomore and junior undergraduates, and assumes only a basic background in Electrical Engineering and Computer Science. We've introduced Jupyter Notebook Python labs to complement the theoretical material covered in more traditional 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—in particular, pure software assignments using 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 the 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 Notebooks, virtual labs, educational technology, signals and systems, electrical engineering.*

1. Introduction

1.1. Signals and Systems at UC Berkeley

Undergraduates in our Electrical Engineering and Computer Sciences (EECS) Department at UC Berkeley take six lower-division courses: (1 and 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 constitute a design and modeling trilogy, and the last three a computation trilogy. Students take these two sequences in parallel during their first four semesters.

Our course in Signals and Systems assumes completion only of the first two courses in each trilogy—constituting a background in linear algebra, basic modeling, and two semesters of programming experience. We cover linear time-invariant (LTI) system theory, Fourier analysis, analog-to-digital sampling theory, and system analysis using the Z and Laplace transforms. Our course is a hub to the upper-division EECS curriculum, including more advanced courses, such as *Digital Signal Processing* and *Feedback Control Systems*.

In fall 2015, our Department's lower-division EE curriculum was restructured, with courses (1 and 2) replacing *EE 40: Introduction to Microelectronic Circuits* and *EE20N: Structure and Interpretation of Signals and Systems*. Our Department used to offer two Signals and Systems courses, one in the lower-division (*EE 20N*) and one in the upper-division (*EE 120: Signals and Systems*, our course). 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 curriculum revision, 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, the 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, the Jupyter Notebook has exploded in popularity as an educational 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 Department, many other courses, such as *EE 123: Digital Signal Processing* and *EECS 126: Probability and Random Processes*, use Jupyter Notebook assignments.

In light of our course's role in the EECS curriculum, we reintroduced a hands-on lab component, and specified three requirements to make better use of the Jupyter Notebook's rich features. 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 proficiency in the modern scientific computing libraries used in industry and academe. Physical labs in a Circuits course teach students the ins and outs of hardware tools, such as oscilloscopes and function generators. Virtual labs in a 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, budget, and other resources exist.

In this paper, we offer a tour of one of our labs. We discuss how—through a delicate interplay of Python code, embedded media, and graphical representations of data—we've created virtual labs that take advantage of the strengths of the Jupyter Notebook to enhance student learning. Counterpart labs that we've surveyed at peer institutions fall short in at least one of these three design principles. Creating labs that satisfy these criteria has allowed us to cover the gamut from basic theory to state-of-the-art applications, and to limit logistical overhead.

2. The Labs

2.1. Related Work

Signals and Systems courses often use labs to reinforce theoretical content from lecture:

- *Intro to Signal Processing* ("ECE 2026," 2018) at Georgia Tech features MATLAB coding labs that extend conventional pen-and-paper problem sets. Most labs have a strong application focus and concrete visual or auditory components, but demand substantive in-person instructor time for verification of multiple checkpoints.
- *Signals and Systems* ("ELE 301," 2011) at Princeton has MATLAB-based labs, several requiring in-person checkoffs of hardware or software implementation.
- *Signals and Systems* (https://sigproc.mit.edu/fall19) at MIT has Python exercises. As the students implement most exercises using Python primitives, they don't taste the richness of Python's open-source scientific computing libraries, such as NumPy and SciPy for multidimensional arrays and signal processing, and Matplotlib for drawing plots.

The 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 intended for classroom use. Furthermore, many are not self-contained. For example, some use mathematical knowledge, such as probability and optimization, without providing sufficient background.

Table 1. Labs for *EE 120: Signals and Systems* at UC Berkeley, Spring 2020.

Lab	Topics
Lab 1: Introduction to Python for Signals and Systems	Essentials of the Python programming language and scientific libraries, rectangular/exponential signal generation, convolution
Lab 2: Applications of LTI Filtering	1D edge detector, simple moving average for denoising, exponential moving averaging of stock price data
Lab 3: Practical Fourier Analysis	Naive Discrete Fourier Transform (DFT), matrix-vector DFT, and FFT implementation, virtual oscilloscope calibration
Lab 4: Heart Rate Monitoring	Spatial averaging of video of patient's thumb for dimensionality reduction, extracting heartbeat frequency through FFT
Lab 5: Deconvolution and Imaging	2D convolution (image blurring and sharpening), deconvolution (audio echo cancellation, Hubble telescope image deblurring)
Lab 6: Control	Closed-loop system analysis, signal filtering, root locus analysis, feedback control of a virtual inverted pendulum

2.2. Summary of Contributions

Table 1 summarizes the Jupyter Notebook labs deployed in the spring 2020 semester. Each Notebook is a single document that consists of text, code, and embedded media "cells" that are rendered or executed. Although teaching staff provide support during office hours and through the Piazza online discussion platform at https://piazza.com, most students complete the labs autonomously and asynchronously 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.
- Test cases and juxtaposed plots of expected and actual results help students verify their progress without divulging the algorithm, reducing debugging frustration.
- Extensive references allow students to dive into the literature that inspired the labs.

3. Case Study: Deconvolution and Imaging 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. By introducing our students to higher-dimensional problems and unexpected contexts, the lab augments their understanding of the operations and transforms taught in the course.

3.1. Real-World Applications

Students often ask, "Can we *undo* convolution?" This lab presents two deconvolution problems. In the first, they remove echoes from corrupted audio data (Eneroth, 2001). In the second, they extend 1D concepts learned in the first problem to deblur 2D image data. Fig. 1(a) shows deep-space image blurring produced by mirror imperfections of the $1.5B Hubble Space Telescope. Inspired by expert proposals of that time (e.g., White, 1992), the lab teaches students how to apply deconvolution algorithms to deblur such images. Fig. 1(b) shows a deblurred image. Students reproduce stunning 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.

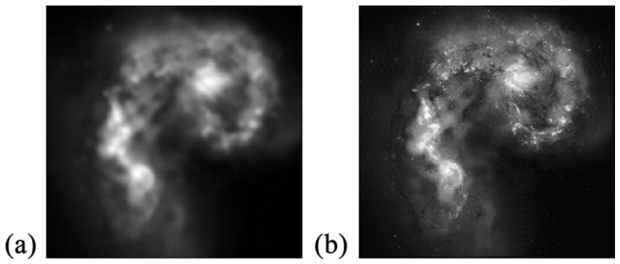

Fig. 1. *Space image deconvolution results. Source: ESA/Hubble, distributed under CC BY 4.0.*

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 module, 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 module, they learn how to use the `matplotlib.pyplot.imread` function to read an image and understand how its data matrix renders on the screen. Fig. 2 illustrates how a 1D integer array is converted to a 2D matrix, and how each matrix entry is mapped to a pixel luminance.

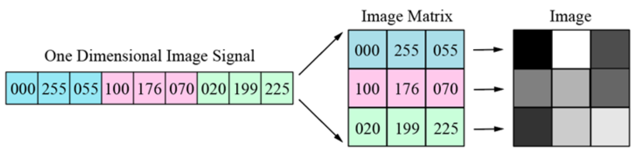

Fig. 2. *Guided example of understanding the underlying image data. Source: Own elaboration.*

Next, the lab shows students how to use, and understand the outputs of, functions from the `numpy.fft`, `numpy.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 that invokes these libraries should take only seconds.

A drawback of functions in high-level scientific computing libraries is that they often accept unwieldy combinations of parameters. Student submissions and feedback helped us find ways to mitigate this. At the first appearance of each function, 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 inject deliberate errors into their code and interpret the results. In this lab, we ask students to blur an image, Fig. 3(a), using a Gaussian filter having an unsuitable convolution parameter, rather than a recommended one. They must then comment on the sudden appearance of a black border in the resultant image, Fig. 3(b). This teaches the effects of parameter modifications, and cultivates debugging skills when students identify similar errors in later exercises.

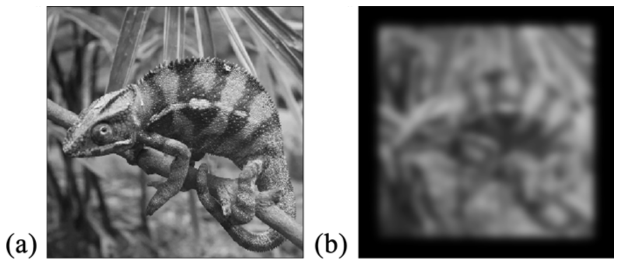

Fig. 3. *Example of purposeful error. Source: Own elaboration.*

Finally, we introduce the parameter sweep, an effective brute force method. Often, there is no simple expression for an optimal parameter value—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.

To address this, the lab guides the 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, the Jupyter Notebook's cell-based design allows for students to complete sections 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. Student Feedback

After each lab, we conduct an anonymous survey to determine what to fine-tune in the future. Fig. 4 depicts how self-contained, interesting, and applications-driven our students found the first three labs offered in spring 2020. Typically, we receive 20-30 responses—about a 25% yield. Our data suggest that a high percentage of our students believe that the labs adhere to the design principles we've laid out in this paper.

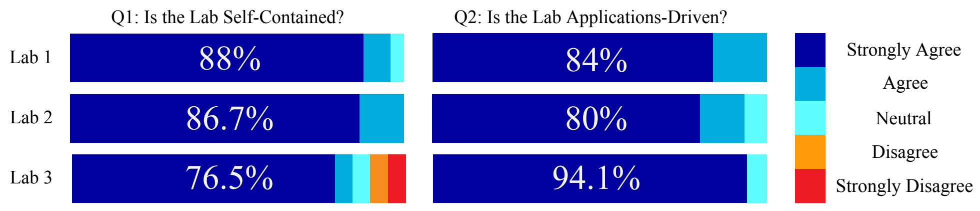

Fig. 4. *Anonymous student feedback.* ***Q1:*** *"The lab was self-contained, and all of the information I needed to complete it was in the instructions or easily Google-able documentation."* ***Q2:*** *"The lab was interesting and applications-driven, with a clear connection between its content and the concepts from lecture and discussion."*

5. Conclusions and Future Work

We've showcased an approach to lab design that emphasizes applications as well as a flavorful use of open-source scientific computing libraries to train and motivate students for research and industry. Every resource needed to complete each lab is embedded therein. So, each lab is self-contained, scalable, and deployable with limited administrative overhead.

We have six labs now, and plan to create four additional ones. We will publish all ten online at https://github.com/dominiccarrano/ee-120-labs, so our colleagues at other institutions can adopt them and give us constructive feedback. We intend to integrate tools for the Jupyter Notebook—such as the *nbgrader* project at https://nbgrader.readthedocs.io—into our labs, so we can automate lab grading in the future.

References

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

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

Eneroth, P. (2001). *Stereophonic Acoustic Echo Cancellation: Theory and Implementation* (Doctoral dissertation). Department of Electroscience, Lund University.

Granado, E. C., & García, E. D. (2018). Guide to Jupyter Notebooks for educational purposes (Á.D. 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.

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. *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). *About Us*. Retrieved April 13, 2020, from https://jupyter.org/about

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. *Astronomical Data Analysis Software and Systems I*, 25, 176-185.