Statement on Teaching & Learning

My vision and goals

As an engineering professor, three main themes underpin my vision and goals on Teaching & Learning in higher eduction: (1) computing embedded in the curriculum; (2) open education; (3) harnessing educational technology, and online and blended learning.

Computing embedded in the curriculum

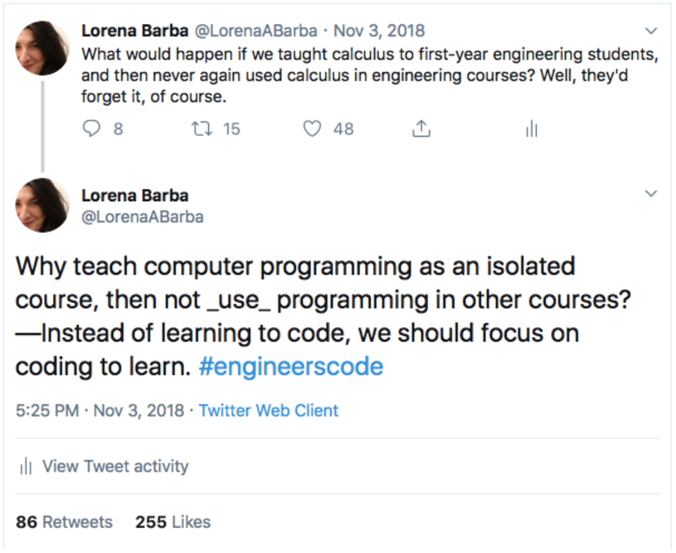
I am a computational scientist, and have taught several engineering courses that incorporate computing. Following the success of two courses in particular—nicknamed "CFD Python" and "AeroPython," more on those later—I came to believe that we can embed computing in almost every engineering course, to fully transform the experience and education of students. Here is a comment I posted on my blog 5 years ago:[1]

> …a better way than the one-semester freshman "intro to programming" course is to embed computing throughout the curriculum, across several courses—engineering math, linear algebra, differential equations, mechanics, heat transfer, fluid mechanics, dynamics and vibration ... all these courses could embed computing as part of their problem-solving approaches. [May 13, 2014]

Engineering education has faced challenges and opportunities of similar transformative power a couple of times before. In the 1990s, engineering colleges began revising curricula to increase attention to *design*, as companies groused that graduates were not ready for productive work.[2] The American Society of Engineering Education[3] recommended that, beyond technical capabilities, engineering schools should provide skills for working on teams, communicating well, and integrating concerns of economic, social, environmental and global contexts. But an even more drastic transformation occurred before, over the first half of the 20th century.[4] Engineering schools used to focus on practical, hands-on skills, faculty were expected to have industrial experience before becoming professors and research was not a usual activity. Around 1920, a shift led by arriving European engineers began placing emphasis on engineering science. After WWII, this focus became more widely adopted, due to new research opportunities from the war effort, an influx of federal money, and large new research facilities. The undergraduate curriculum was forever altered, following the European approach, with less machine shop and surveying, and more science and math courses.[5] These changes in engineering education took hold quickly. Eventually, they led to a divide between engineering schools and industrial practice, which fueled the efforts in the 1990s for curriculum reforms.

In the 21st century, the backdrop for professional engineers has markedly changed. New calls for re-examining engineering education[6] highlight the emergence of the Internet as a reason for knowledge not being "owned" by experts anymore. New engineers need to have the ability to find information quickly, to transform information into knowledge, and to work effectively in global inter-disciplinary teams. But the last decade has seen ever more radical changes. The American Society of Mechanical Engineers found in a 2011 study[7] that among more than 1200 survey respondents, the top most-needed skills cited were: communication skills, and computer/software skills (p.18). Younger respondents gave even greater weight to computer programming as a needed skill. At the same time, a national narrative for "learning to code" has centered on jobs and producing skilled workers. For example:[8] "At a time when people are saying 'I want a good job; I got out of college and I couldn't find one,' every single year in America there is a standing demand for 120,000 people who are trained in computer science" (President Bill Clinton). "Code has become the 4th literacy. Everyone needs to know how our digital world works, not just engineers" (Mark Surman, Mozilla Foundation). "Whether we're fighting climate change or going to space, everything is moved forward by computers, and we don't have enough people who can code" (Sir Richard Branson).

Calls for learning to code are well-intentioned and worthwhile, but miss a key point: the goal is to educate students to enter the workforce in command of computational tools and techniques to solve problems and create new knowledge. In other words, what they need, rather than learning to code, is *coding to learn*. This is my motto, and underpins several of my educational efforts.


Lorena Barba @LorenaABarba · Nov 3, 2018
What would happen if we taught calculus to first-year engineering students, and then never again used calculus in engineering courses? Well, they'd forget it, of course.
8 15 48

Lorena Barba
@LorenaABarba

Why teach computer programming as an isolated course, then not _use_ programming in other courses? —Instead of learning to code, we should focus on coding to learn. #engineerscode

5:25 PM · Nov 3, 2018 · Twitter Web Client

View Tweet activity

86 Retweets 255 Likes

https://twitter.com/LorenaABarba/status/1058772169466044416

In 2017, I obtained competitive funding from NSF for a project (ongoing) to develop reusable, open learning modules for teaching computing embedded in the engineering curriculum. (See "Engineering Computations," below.) Just like engineering education was wholly transformed in a short time frame during the last century, it can be transformed again by embedding computing across the curriculum, with the long-term vision of computing becoming *infrastructural* in the education of STEM graduates.[9] For example, the subject of calculus is infrastructural in engineering education: all learners study it, and further learning relies on it. Being infrastructural is the defining feature of a *literacy*: a socially widespread deployment of skills and capabilities that become material support to achieve valued intellectual ends.[10] The phrase "computational literacy" is often used with a shallow meaning of simply knowing your way around computers and being able to use some standard desktop applications. Andrea diSessa, professor of education at UC Berkeley, developed the alternative deep idea. Conventional literacy, that is, knowing how to read and write, is highly valued in our society and is utterly essential to education: not just a result of education, but its driving force. Computing, claims diSessa, is the basis of a new literacy, empowering the transformation of science education. This is precisely my vision: computing can lead to faster or deeper learning and discovery; but to achieve this promise, we need infrastructural adoption.

I am currently working on the development of open educational materials implementing the vision that engineering students should begin computing immediately upon the start of their degree. They would continue to encounter computing, in context, in their subsequent engineering courses, enabling them to professionally use technical computing throughout their engineering career. With the current explosion of machine learning and artificial intelligence, the need is more critical for all STEM graduates to have a solid foundation in *computational thinking*. This term also suffers from sometimes shallow definitions, but the deep idea from visionary scientist and educator Seymour Papert is that students "can learn to program and it can affect the way they learn everything else."[11] I am following the latest interpretation that defines computational thinking focusing on its application to mathematics and science.[12] The definition is based on a taxonomy in four categories: (1) data practices; (2) modeling and simulation practices; (3) computational problem-solving; and (4) systems-thinking practices. My series in "Engineering Computations" (see below) pursues this interpretation.

Open education

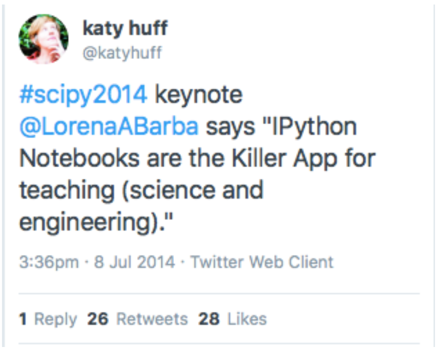
The Open Education movement was inspired by free and open source software (FOSS), but in my opinion has missed on some key features of FOSS: the open development model, networked collaboration, community around open-source projects, the culture and value-based framework. A key point here is that "free" in FOSS is about freedom, not price. In open-source software development, we cherish what some call our *productive freedom*—the freedom to labor in a framework of our own making, side-stepping the restrictions of copyright law by cleverly attaching a license to our work, so that we can prioritize access, distribution, collaboration.[13]

The most visible manifestations of open education are the open courseware (OCW) initiatives, and spread of open educational resources (OER). In 2001, MIT launched its OpenCourseWare initiative, promising free public access to their course materials for noncommercial uses. It was a unique commitment at an institutional level, strengthened by the MIT brand. Other universities joined the OCW movement: Rice with the Connexions project (now OpenStax), CMU with the Open Learning Initiative (OLI), Utah State University with the Center for Open and Sustainable Learning, and so on. A UNESCO Forum coined the term "open educational resources" in 2002, and the Organization for Economic Co-operation and Development (OECD) issued a study in 2007 of the movement and the challenges it faced and represented for higher education.[14] The recurring topics in the OER conversation are: reducing the cost of textbooks for students, increasing access (for worldwide learners), copyright and licenses, altruism and public good. Despite growing anxiety over the high cost of textbooks (a particularly US-centric concern), change to the status quo has been scant. A survey of over 3,000 faculty reported that more than half (58 percent) said they were not aware of OER or how instructors can use free or inexpensive alternatives to traditional textbooks in their courses.[15] The OER narrative is often about creation vs. adoption, author vs. user… and that leads us to a problem: "We create huge amounts of OER, but there is very little reuse," says philosopher and higher-education pundit Stephen Downes.[16] And in a recent short video discussing the importance of openness and the role it plays in knowledge, communication and learning, Downes says this:[17] "Openness is about the possibilities of communicating with other people. It's not about stuff, what you do with stuff. It's about what you do with each other." He's hinting at the key social interactions that make learning meaningful. OER have not been transformative, and one of the reasons in my opinion is the failure to widely adopt the ethics and practice of the open-source culture.

I have published OER for every course that I've taught in the last decade, and even though I often get messages of gratitude from self-learners around the world, only rarely have I been contacted by another instructor interested in reusing these materials in their teaching. My conclusion is that if we want to increase reuse, we need to encourage collaboration and develop *in the open*. Building community should be an active area of investment in our open educational projects, mirroring the culture and collaboration rituals of open-source software. This idea is reflected in my recent efforts with the "Engineering Computations" series, and previously with the MOOC "Practical Numerical Methods with Python."

- A complete list of my published OER, talks, essays and blog posts on open education is given in my statement of Contributions to Open Education, https://doi.org/10.6084/m9.figshare.8320619
- I created video collections for all my courses at Boston University. These collections quickly rose to the top-downloads in the Engineering category (for all institutions on iTunes U) and I became the university's single top provider of educational media on BU's channel.[18] Analytics of the service from January 2016 showed that my content still accounted for nearly 90% of all the channel downloads in a 5-year period.[19]
- In 2014, I self-produced the first massive open online course (MOOC) of the George Washington University, titled "Practical Numerical Methods with Python." Early on, I enlisted two other instructors teaching similar courses in other universities internationally to adopt the course, and contribute peer review of the materials. The course ran again in 2015 and 2016, collecting more than 8,000 online registrations in our self-hosted Open edX platform. The platform was re-installed in 2017, and the course reset; it slowly has climbed back to more than 1,500 registrations, without more dissemination.[20]

I helped launch a new genre of OER with the "CFD Python" collection of IPython notebooks (2013). This set of lessons—consisting of media-rich content intermixed with Python code that can be interactively executed—was widely commented on various online forums and within the Python community. In 2014, I was invited to keynote at the Scientific Python conference, where I declared IPython Notebooks (later re-branded as Jupyter) a "killer app" for STEM education. Many instructors took inspiration from my work when adopting Jupyter for open courseware and tutorials, and publishing their materials in the open model. Today, we can find a cornucopia of tutorials, lessons, courses and even whole books created in Jupyter.[21] I later coined the term *computable content* for this new genre of OER, defining it as "educational content made powerfully interactive via compute engines in the learning platform." My work and


katy huff
@katyhuff

#scipy2014 keynote @LorenaABarba says "IPython Notebooks are the Killer App for teaching (science and engineering)."

3:36pm · 8 Jul 2014 · Twitter Web Client

1 Reply 26 Retweets 28 Likes

https://twitter.com/katyhuff/status/486519116451176448 conceptualization inspired a new wave of educational publishing by the company O'Reilly, which publicly credits me. See, e.g.:

> "Computable content: Notebooks, containers, and data-centric organizational learning," by Paco Nathan (O'Reilly Media), December 7, 2016. https://conferences.oreilly.com/strata/strata-sg-2016/public/schedule/detail/54253

Our MOOC in 2014 took the concept further: not only were the course materials open (released under a permissive license), but I also built an online course on a self-hosted platform. I used the software created by MIT and Harvard for the edX start-up, which had been recently released as open-source (Open edX). At the time, universities were rushing to partner with third-party vendors to jump on the bandwagon of MOOCs. I explained my goal in self-hosting as:[22] "to offer a MOOC without surrendering our IP to for-profits or subjecting students to creepy data mining." I showed how a university, an academic department, or even an individual faculty member, could offer an open online course independently, without involving a commercial platform. The idea was ahead of its time in 2014, but now it is becoming increasingly clear that partnerships with for-profits are not in the best interest of learners. Our course itself contained effectively a tutorial on the open-source ethic. A learner reviewing the course said:[23] "Practical Numerical Methods With Python doesn't just teach its students numerical methods—it also teaches them how to be good scientists and netizens. Open-source and scientific ethics permeate the course structure." In essence, the course's goal was forming a community. This design shows a vision beyond OER, to open educational networks where people and their interactions are more important than content.

Harnessing educational technology, online and blended learning

I began innovating using technology to support learning since my first faculty appointment in 2004, first producing live lecture screencasts using a whiteboard capture system and screen-recording software. After this, I adopted a method of instruction using digital inking with a graphic tablet accessory for on-screen annotations. I recorded most of my lectures at BU using this method to create my iTunes video collections. Later, I distributed the previously recorded screencasts as pre-class material, to focus classroom time on embedding knowledge with active learning. Adopting this *flipped classroom*[24, 25, 26] approach was part of the educational program of my NSF CAREER proposal, and I used it in my Computational Fluid Dynamics course in 2012. It is now a broadly accepted teaching format, but was all but unknown when I adopted it. The successful experience was covered by the College news[27] and I later gave several talks about the flipped classroom. Given my recognized experience with flipped learning, I was invited to be guest editor of a special issue of *Advances in Engineering Education*:

> "Guest editorial: Flipped classrooms in STEM," Barba, Lorena A., Kaw, Autar, LeDoux, Joseph M.. *Advances in Engineering Education*, 5(3) (November 2016), American Society for Engineering Education. http://advances.asee.org/publication/guest-editorial-flipped-classrooms-in-stem/

The first book on flipped learning in higher education appeared in mid-2017. It covered my work on pages 18–19. See: *"Flipped Learning: A Guide for Higher Education Faculty"* by Robert Talbert. Link to Amazon: http://a.co/9ACjd6q

Although online learning often relates to distance education, it can also be used to improve learning in the traditional classroom. Efforts to bring online learning to the on-campus experience are alternatively called hybrid learning or blended learning (the flipped classroom is a form of blended learning). My Fall 2014 MOOC (a.k.a. Numerical MOOC) ran at the same time as my on-campus class on numerical methods, so I could both use the online course to enhance the face-to-face learning experience, and also give my students the chance to participate in a community of learners around the world. In Spring 2015, I built an online course in our Open edX platform (https://openedx.seas.gwu.edu) to supplement my on-campus Aerodynamics course, using the AeroPython lessons (even though I did not advertise it as a MOOC, it is open for anyone to enroll). This allowed me to add auto-graded student assignments using the built-in problem types of Open edX, with students entering answers obtained by running code outside the platform. It also helped organize learning sequences for the students, pointing to the outside material (the AeroPython lessons on GitHub), and peppering with complementary videos and conceptual quizzes as formative assessment.

For the new series in "Engineering Computations" (see below), I am creating online "courses" for shorter content modules (equivalent to about one-third of a course, or one university credit), and using more advanced auto-grading based on Jupyter. Modularization was one of the defining features of Numerical MOOC: it is built of five modules, but they were collected in one online course (each module making a section). The idea of unbundling the course into short modules has the double goal of motivating students with partial achievements, and facilitating adoption by other instructors who can build their courses using a "mix-and-match" method.

Through these experiences, I have refined an approach I call *Jupyter-first* course development: making a course first as a set of Jupyter notebooks, then building both an online course and an on-campus learning experience based on those notebooks. With outside technical partners, we developed third-party extensions for the Open edX platform (called XBlocks) to integrate both content and assessments based on Jupyter into an online course. The Jupyter Viewer Xblock allows constructing learning sequences with content pulled dynamically from a public Jupyter notebook (e.g., on GitHub). We also developed an integration to use the nbgrader[28] Jupyter extension to auto-grade assignments prepared in Jupyter, within Open edX. These new developments (released as open source) were presented in:

- Jupyter-based courses in Open edX: Authoring and grading with notebooks, Open edX Conference, May 2018, Montréal, Canada. Slides: https://doi.org/10.6084/m9.figshare.6553550
- Engineers Code: re-usable, open educational modules for engineering undergraduates, SciPy Conference, July 2018, Austin, Texas. Slides: https://doi.org/10.6084/m9.figshare.6818279

Open edX is a full-featured open-source platform for online courses, used by millions of learners via the edX consortium, large MOOC platforms abroad (France, China, Spain, and others), and institutional deployments (https://open.edx.org). Jupyter-first courses can be written outside the course platform, using an open development model (like any open-source software project), collaboratively and under version control. Once the material is written, one can build a MOOC-style course on Open edX, pulling the content from the notebooks without duplication in the course platform. (Note that Open edX has no concept of version control.) One can interleave short videos and graded sub-sections using the built-in problem types, or using our Graded Jupyter XBlock. This course development workflow is the product of several years of refinement, and applies evidence-based instructional design. Combined with modern pedagogies used in the classroom, like active learning via live coding, one can create learning experiences that are effective both on campus and online.

Courses using computing (with Python and Jupyter)

CFD Python

"CFD Python" (a.k.a., "the 12 steps to Navier–Stokes"), written in 2013, is a learning module based on a set of exercises I had been using for several years as part of an upper-level engineering course in computational fluid dynamics. The module places emphasis on learning by first programming solutions to simple models, and building up to more complex models. With one of my students, we wrote the lessons as a set of Jupyter notebooks in 2013, and published them openly on GitHub. We announced the release on the group website:[29]

> Prof. Barba has a long track record of sharing educational materials freely online. Recordings of her lectures for the CFD course were made available first on iTunesU in 2010. Then, in 2012, she decided to use the flipped classroom model and re-posted the videos on YouTube, after editing for length and with some additions. The new CFD Python class notebooks are her latest free online materials!

We recently published a paper about the module in:

- Barba, Lorena A., and Forsyth, Gilbert F. (2018). CFD Python: the 12 steps to Navier-Stokes equations. *Journal of Open Source Education*, 1(9), 21, https://doi.org/10.21105/jose.00021

The article includes a summary of the module, a statement of need, some background on how it has been used in the classroom, examples of adoption by other instructors, and some notes on instructional design. The CFD Python module is often recommended on sites like *CFD Online*, and *Quora*, was translated to Spanish by a volunteer,[30] and has been adopted by other instructors around the world.[31]

AeroPython

In Spring 2014, I was tasked with teaching a classical aerodynamics course for seniors and first-year graduate students. As I explained in a later blog post:[32] "I asked myself: what is the one thing that I want students to take from a classical Aerodynamics course? The *one* thing? My answer was the use of potential flow for aerodynamic analysis, via the panel method." AeroPython is a series of lessons that begin from the simplest concepts in potential flow, and take students in a step-by-step fashion to build their own 2D panel-method code for lifting bodies. We published a paper about the open course in:

- Lorena A. Barba and Olivier Mesnard (2019). Aero Python: classical aerodynamics of potential flow using Python. *Journal of Open Source Education*, 2(15), 45, https://doi.org/10.21105/jose.00045
- GitHub repository: https://github.com/barbagroup/AeroPython

Practical Numerical Methods with Python

Launched in Fall 2014 both as the first MOOC in GW and as an on-campus course, this is a first course in numerical methods for differential equations aimed at engineering and science students. The materials are presented as a set of five discrete learning modules that build on each other, each consisting of four or five lessons written as a Jupyter notebook, and a student assignment. The MOOC was set up to award open digital badges for each completed module.[33] More than 8,000 people ended up enrolling for the open course over about three years (until we refreshed the online platform with a new version of Open edX, and reset the student roster).

- Numerical MOOC: Collaborating in Open Education for CSE, poster presented at the SIAM Conference in Computational Science and Engineering 2017, Atlanta, GA, https://doi.org/10.6084/m9.figshare.4696384
- GitHub repository: https://github.com/numerical-mooc/numerical-mooc

Engineering Computations

The "Engineering Computations" project is building a series of learning modules in technical computing for undergraduate students in engineering and science. It started in Fall 2017, supported by NSF under the OAC CyberTraining program. Our goal is to make open instructional materials that are designed to be reusable, and embark in community efforts to share good practices for teaching with them. On campus, I have twice taught a sophomore course with three beginner modules written in 2017. This course aims to create a foundation in computational thinking for engineers and to develop confidence in using *coding to learn* in various technical contexts.

- Course syllabus available online at: https://doi.org/10.6084/m9.figshare.5709748.v1
- Engineers Code: Re-usable, Open Educational Modules for Engineering Undergraduates, poster presented at JupyterCon 2018, New York, https://doi.org/10.6084/m9.figshare.7150952
- GitHub repository: https://github.com/engineersCode/EngComp
- Three separate modules in the Open edX platform https://openedx.seas.gwu.edu
  - http://go.gwu.edu/engcomp1
  - http://go.gwu.edu/engcomp2
  - http://go.gwu.edu/engcomp3

In 2018, my department in GW approved a two-course sequence, teaching two beginner modules in the freshman year, and writing more advanced material for the sophomore year. In addition, we will write “add-on” modules that can be used to support other engineering courses. This is work-in-progress (all being developed openly on GitHub).

Pedagogy and instructional design

Some key concepts and design principles that are distilled from several years developing the courses described above are:

1. the idea of computable content—educational content made powerfully interactive via compute engines in the learning platform—using Jupyter;
2. the idea of open pedagogy: reflecting in the teaching practice the ethos and practices of open source software;
3. modularization: creating stackable learning modules that break-up the standard “course” format;
4. harnessing the worked-example effect, empirically shown to be superior to problem-solving for novice learners;
5. using live-coding to structure active-learning class experiences;
6. guiding learners to document their own work, also on Jupyter.

Reading the education-research literature, I have come to understand some of the principles that come into play to design effective computational modules. Active learning has been shown to be superior to lectures in both comprehension and recollection,[34] and using Jupyter notebooks combined with techniques like live coding, pair programming, and others, promotes active learning. The worked-example effect,[35] the best known and most widely studied of the cognitive load effects, explains why our design helps students manage the complexity of learning applied computing. Modularization, chunking, interleaving… several known effective pedagogical techniques can be put to work in this model.

Qualifications, Awards

At the University of Bristol, UK, I completed the Teaching and Learning in Higher Education program[36] in 2007 (a four-module course including a Teaching Portfolio). In 2008, I was nominated and awarded the “Rising Star” teaching award,[37] recognizing an early career faculty member for “the quality of reflection, analysis and practice” in her teaching.

In 2012, I was nominated by the Boston University Dean of Engineering and then selected to participate (fully funded) in the National Academy of Engineering (NAE) Frontiers of Engineering Education Symposium.[38] NAE presents the selected participants as the “nation’s most engaged and innovative engineering educators.”

In December 2016, I was awarded the Leamer-Rosenthal prize in Open Social Science by the Berkeley Initiative for Transparency in the Social Sciences (BITSS) in the “Leaders in Education” category.[39] And in February 2017, I learned that I had been nominated and I received an

Honorable Mention in the Open Education Awards for Excellence of the Open Education Consortium.[40]

This year, I was a finalist for the 2019 Morton A. Bender Teaching Award at the George Washington University.

References

[1] https://lorenabarba.com/blog/why-i-push-for-python/#comment-1383694622

[2] Masi, C.G., 1995. Re-engineering engineering education. *IEEE Spectrum*, 32(9), pp.44-47. https://doi.org/10.1109/6.406465

[3] ASEE, 1994. Engineering Education for a Changing World (Green Report), https://www.asee.org/papers-and-publications/publications/The-Green-Report.pdf

[4] Seely, B.E., 1999. The other re-engineering of engineering education, 1900–1965. *Journal of Engineering Education*, 88(3), pp. 285-294. https://doi.org/10.1002/j.2168-9830.1999.tb00449.x

[5] ASEE, 1968, Goals of Engineering Education, https://www.asee.org/documents/publications/reports/goals_of_engineering_education.pdf

[6] Tryggvason, G. and Apelian, D., 2006. Re-engineering engineering education for the challenges of the 21st century. *JOM Journal of the Minerals, Metals and Materials Society*, 58(10), pp.14-17. https://doi.org/10.1007/s11837-006-0194-6

[7] ASME (2012), The state of Mechanical Engineering: Today and Beyond, https://www.asme.org/wwwasmeorg/media/ResourceFiles/Campaigns/Marketing/2012/The-State-of-Mechanical-Engineering-survey.pdf

[8] From: https://codebeedo.com/what-did-they-say-about-coding/

[9] Quoting from our grant proposal, shared publicly as: Barba, Lorena A.; Wickenheiser, Adam; Watkins, Ryan (2017): CyberTraining: DSE—The Code Maker: Computational Thinking for Engineers with Interactive, Contextual Learning. *figshare*. https://doi.org/10.6084/m9.figshare.5662051.v1

[10] DiSessa, A.A., 2001. Changing minds: Computers, learning, and literacy. MIT Press.

[11] See also my essay "Computational Thinking: I do not think it means what you think it means," https://medium.com/@lorenaabarba/computational-thinking-i-do-not-think-it-means-what-you-think-it-means-6d39e854fa90

[12] Weintrop, David, et al., 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25(1): 127-147. https://doi.org/10.1007/s10956-015-9581-5

[13] Coleman, E.G., 2012. Coding freedom: The ethics and aesthetics of hacking. Princeton University Press.

[14] OECD, 2007. Giving Knowledge for Free: The Emergence of Open Educational Resources, http://doi.org/10.1787/9789264032125-en

[15] Opening the Textbook: Educational Resources in U.S. Higher Education, 2015-16 , I. Elaine Allen and Jeff Seaman, Babson Survey Research Group (July 2016), http://www.onlinelearningsurvey.com/reports/openingthetextbook2016.pdf

[16] Stephen Downes, 2010. The Role of Open Educational Resources in Personal Learning, VI International Seminar of the UNESCO chair in e-Learning, https://youtu.be/AQCvj6m4obM

[17] Stephen Downes, 2017. https://youtu.be/FPHYAFcUziA

[18] See: http://lorenabarba.com/news/bus-top-provider-of-educational-media/

[19] See the slide deck *"A Pathway in Open Teaching"* https://speakerdeck.com/labarba/a-pathway-in-open-teaching DOI: 10.6084/m9.figshare.2068182.v1 (January 2016).

[20] Online course: https://openedx.seas.gwu.edu/courses/course-v1:MAE+MAE6286+2017/about (site updated in 2017).

[21] https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

[22] https://lorenabarba.com/news/announcing-practical-numerical-methods-with-python-mooc/

[23] https://www.classcentral.com/report/gwu-numerical-mooc-review/

[24] Pink, D., 2010: Think tank: Flip-thinking — the new buzz word sweeping the US. *The Telegraph*, http://www.telegraph.co.uk/finance/businessclub/7996379/Daniel-Pinks-Think-Tank-Flip-thinking-the-new-buzz-word-sweeping-the-US.html

[25] Martin, J., 2010: Reverse instruction: Dan Pink and Karls "Fisch Flip". *Connected Principals*, http://www.connectedprincipals.com/archives/1534

[26] Bergmann, J., J. Overmyer, and B. Wilie, 2011: The flipped class: myths vs. reality. *The Daily Riff*, http://www.thedailyriff.com/articles/the-flipped-class-conversation-689.php

[27] See: https://www.bu.edu/eng/2012/03/08/flipped-classroom-energizes-computational-fluid-dynamics-course/

[28] http://nbgrader.readthedocs.io/en/stable/

[29] https://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

[30] https://github.com/franktoffel/CFDPython-ES

[31] For example: http://cav2012.sg/cdohl/CFD_course/

[32] https://lorenabarba.com/blog/announcing-aeropython/

[33] See: https://lorenabarba.com/blog/a-collaboration-to-issue-badges-in-numericalmooc/

[34] Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences, 111(23), 8410-8415.

[35] Sweller, J. (2006). The worked example effect and human cognition. Learning and instruction. 16(2), 165-169, http://dx.doi.org/10.1016/j.learninstruc.2006.02.005

[36] The program is now superseded, but see the Internet archive at https://web.archive.org/web/20081014200110/http://www.bristol.ac.uk/esu/tlhe/

[37] The "Rising Star" program is now defunct, but see the Internet archive at https://web.archive.org/web/20070211144015/http://www.bris.ac.uk/tsu/awards/prizes/risingstar.html

[38] See: www.nae.edu/Projects/CASEE/26338/69844/20742.aspx

[39] See (and links therein): http://lorenabarba.com/news/prof-barba-awarded-a-2016-leamer-rosenthal-prize-for-open-social-science/

[40] See http://www.oeconsortium.org/projects/open-education-awards-for-excellence/honorable-mentions/