CyberTraining: DSE—The Code Maker: Computational Thinking for Engineers with Interactive, Contextual Learning

Lorena A. Barba, Adam M. Wickenheiser and Ryan Watkins , The George Washington University.

Introduction

We want to transform how engineering students learn computing, increasing both their achievement in and their integration of computing as a career-enabling skill. We propose to design, develop and deploy a series of learning modules that

1. Are context-based and immediately applicable, integrated in the curriculum;
2. Use a high-level language (Python), and state-of-the-art tools;
3. Are delivered just-in-time, supported both in-person and by an online platform.

Students will complete the modules over the course of their degree, supporting the discipline courses. They will be built around core engineering material so that they may be adopted fully or in part at other universities.We will work with course instructors to develop relevant context for the modules, to coach them in the new material, and to coordinate how and when they require or encourage completion of the modules (and provide students time-management guidance). The modules will be supported by an open lab and tutorials with trained learning assistants (LAs). Support activities will be held in a new Gelman Library facility, known as "STEM Lab" (GW Today, Oct. 30, 2016), where the space and activities will be inspired by the Maker Movement (e.g., "Hack Space" and "Code Maker").

We will partner with colleagues at other universities who will offer peer review of the materials, will trial them in their local environments, and will help build a community around the educational materials and pedagogical methods of this project. (See letters of collaboration.) We also have leaders from industry who will act as informal advisors through the progress of the project, and offer authentic scenarios of computing skills required of interns and employees.

All educational materials will be open (under CC-BY and BSD 3-clause licenses), and we will deploy cloud infrastructure for open online learning, using the Open edX platform. For local, on-campus use, we will also deploy private cloud infrastructure for computing using JupyterHub (see details in the Technology section of this proposal). To facilitate adoption of these tools, we will publish tutorials, technical briefs, and conversational blog posts describing our setup and activities.

Long-term, the project will not only train our cohort of students in employment-ready computational skills, but will build community to disseminate open educational materials and know-how for integrating computational infrastructure in the undergraduate curriculum. The local cohort of students stands out for gender balance, as GW ranks #5 nationally in the percentage of female engineering students (now, 40%). Recently, SEAS committed to a drive to become #1 (see @GWEngineering tweet, 11:01 AM 23 Nov 2016). This project will strengthen the gender balance by purposefully recruiting females to train as learning assistants for the STEM Lab, and coaching them in peer-mentoring techniques. Our target is a 50% participation rate of women in the LA team.

1. Workforce challenge addressed

This project is designed for training undergraduate engineering students to ready them for joining the workforce with solid computational skills, for going into PhD programs in computational science, and for becoming sophisticated cyberinfrastructure users.

Based on the experience of the PI as leader of a research group in computational science, with a long track record in open education and public involvement in the computational-reproducibility movement, the training program will get students ready to join any similar research group.

With input from industry advisors—from, for example, Boeing and GE Global—the program will give students solid preparation to join internships and future careers that use cyberinfrastructure.

The need for this preparation is clear. Advances in computing are "a key driver of economic competitiveness," says the report of the President's Council of Advisors on Science and Technology "Designing a Digital Future" (Dec. 2010). They are "crucial in achieving our major national and global priorities" in energy, education, healthcare, national security, and more. And they "accelerate the pace of discovery in nearly all other fields." The US Bureau of Labor Statistics projects that the top-five STEM careers with the most growth are all in computing (Adams, 2014). Meanwhile, nearly 50% of H1-B visa petitions approved in fiscal year 2009 were for computer-related workers (US Dept. of Homeland Security, April 2010). Computing is a hot job market.

At the college level, the good news is that the number of computer-science graduates has been increasing in the last few years, after an 8-year slump ending in 2009 (CRA, 2013). The bad news is that 30% to 50% of students fail or withdraw from their introductory CS class (Porter et al., 2013). In other STEM majors, the percentages could be higher, since these students did not choose computing as their major. It is well known that engineering students often do not use computing effectively as a learning tool, do not "like" their introductory classes (reflected in our own student surveys), and do not develop computational literacy as part of their undergraduate education.

2. We need to transform how we teach computing

Evidence shows that even CS majors scarcely improve their computing skills after the introductory courses. A classic Yale study (Soloway et al., 1983), using the now-famous "rainfall problem," had shocking results: only 14% of beginning students could solve the problem after CS instruction, and only 36% of junior students succeeded. This study has been repeated many times, with equally appalling results (Guzdial, 2010). A multi-institutional, multi-national study with 216 students testing on a language-independent computing test resulted in just a 21% success rate (McCracken et al., 2001).

"Why is it so hard to learn to program?"

After surveying several studies, Guzdial (2010) explains why introductory programming courses have such low success rates: in natural-language task descriptions, people don't define iterations, but set operations instead, they are not explicit about loops; people use constraints, event-driven tasks and

imperative programming, but they never talk about objects. So, an introductory computing course often presents students with knowledge that contradicts their mental models. And lecture-based courses in computing have low efficacy. Teaching computing to beginners needs to shift the emphasis to the learner, providing relevant activities that help students construct new algorithmic models.

This project is based on a learner-centered premise that to make learning more effective, students need to construct their knowledge. If not, and if the knowledge we want them to acquire contradicts in any way their mental models, students can merely memorize without understanding. In that case, students will not apply the new knowledge to solve problems in the future.

Context-based computing

Many discipline experts argue for context-based computing. A recent report of the National Research Council (2011) on pedagogical aspects of computational thinking says that … the power of computational thinking is best realized in conjunction with some domain-specific content." Applying this philosophy, Magana, Falk & Reese (2013) created a freshman course on computation and programming for materials scientists and engineers at Johns Hopkins University, designed to be learner-centered. The evaluation found that the course succeeded in increasing students' perceptions of ability and their recognition of the importance of computing in their fields. The Georgia Tech experience with a media-computing course also shows that context-based teaching of programming increases student motivation and success (and in their case, it also reduced the success gender gap).

Students experience frustration when they do not immediately see the usefulness of what they are learning (and often struggling with) in a first programming course. Successful initiatives to improve introductory computer science often implement "programming in context." For example, the changes introduced in Harvey Mudd College to increase not only general retention but also student diversity included: replacing the programming language (from C to Python) and introducing applied problems with interesting contexts. Their success has been outstanding on all fronts (Hafner, 2012). At the high-school level, Beaver Day Country School began integrating coding projects into a wide array of subjects in lieu of a separate programming course. "We also recognize that coding is a mindset, so we don't want our students to memorize a certain list of commands within a certain programming language. Instead, we want them to think about solving problems in innovative ways," says Rob MacDonald, Beaver Day Country School's Math Department Head (Larson, 2013).

3. "The Code Maker" training program

We will develop a program of 8 or more learning modules, each applying computing to relevant situations in engineering. A typical module will contain four lessons as a Jupyter Notebook and will be designed to be completed in 3 to 4 hours. A student may complete a module over a relaxed two- or three-week stretch, or "binge" over a weekend (say). Each module will end with an assignment, for the student to apply the computational skills in a similar situation to the lessons, but with variations that test conceptual thinking. The modules will embed computing in the existing curriculum, with the long-term vision of computing becoming infrastructural in the education of STEM graduates.

In recent years, the PI's group has produced similar learning modules at the graduate level: **"CFD Python"** and **"AeroPython"** are the first examples. The five modules of the on-campus course and MOOC **"Practical Numerical Methods with Python,"** a.k.a. Numerical-MOOC, have been used by hundreds of learners around the world, and in-class for three semesters. (See References for links.)

The "Code Maker" training program will be locally supported by a physical space in the Gelman Library ("STEM Lab"), and a team of specially trained LAs. The GW Learning Assistant program follows the Colorado-Boulder Model and LAs complete over 30 hours of specialized training, meeting all three requirements of that program: practice facilitating interaction and discourse, comprehensive pedagogical training, and weekly meetings with course faculty. The PI, co-PIs, graduate student supported in this grant, and the LAs will hold regular activities to both support the structured learning with the modules and to create a social environment to make computing fun and relevant.

The learning experience provided by the modules will aim to develop not just programming skills but computational thinking, in context. The goal is to prepare students to use computing in later classes to solve problems and develop confidence and computational literacy that will open doors for employment, undergraduate research or graduate study.

Computational thinking

The Code Maker program aims to prepare engineering undergraduates to enter the workforce (or graduate programs) with command of computational tools and techniques to solve problems and create new knowledge. We associate this with computational-thinking skills, not just programming.

Computational thinking has a buzzword quality and is often misunderstood (Barba, 2016). The popular meaning is that it refers to an "attitude and skill set" (Wing, 2006) for solving problems using concepts from computer science: decomposition, abstraction, pattern recognition, algorithms. This reductionist explanation leaves out the essence, as envisioned by its originator, Seymour Papert: that "computers might enhance thinking and change patterns of access to knowledge" (Papert, 1980). Papert was a pioneer of artificial intelligence (co-director of the MIT AI Lab with Marvin Minsky) who previously worked with developmental psychologist Jean Piaget. He dedicated decades to studying how children learn and the role of technology in education, and invented the LOGO language. He thought that computers can improve our understanding of thinking and learning, they can give us access to powerful ideas and complex concepts that are difficult to grasp with our unaided mind. He is often quoted: "You can't think about thinking without thinking about thinking about something." (e.g., NPR, Aug. 2016; MIT Media Lab, 2016a) His collaborator Nicholas Negroponte clarified that "…the act of programming was a very good approximation of thinking about thinking. " (MIT Media Lab, 2016b).—"The goal is to use computational thinking to forge ideas" (Papert, 1996).

The Code Maker training program will coach undergraduate engineering students to adopt computing as a powerful instrument of learning and discovery. To do it, we take inspiration in Papert (also known as the father of the Maker movement), build on our previous experience, and incorporate best practices of pedagogy and instructional design.

Why Python

Quoting Bertrand Meyer (2003): A good teaching language should be unobtrusive, enabling students to devote their efforts to learning concepts," not syntax. Mannila et al. (2006) analyzed novice-written programs in Python, Java and C++. They found far fewer errors in Python (thanks to its simple syntax), a boon when teaching novices who get frustrated by syntax errors. They also found fewer logic errors, and the code was more structured (in part, thanks to Python's indentation). And when students transitioned to a more complex language, there was no handicap due to having learned in Python.

In Mannila & de Raadt (2006), the following criteria are offered for a language to be used in an introductory programming course:

- it is designed with teaching in mind (simple syntax, natural semantics)
- it can be used to apply physical analogies (provides multi-media capabilities)
- it offers a general framework (serving as a basis for learning other languages later)
- it promotes a new approach for teaching (augmented by principles, tools and libraries)

With additional criteria involving the design environment (e.g., interactivity), support and availability, these authors rated 11 popular languages, and concluded the best languages for teaching are Python and Eiffel, followed closely by Java. A decade later, however, an item where Python lagged behind —providing a seamless development environment—has been corrected. Now, we not only have various full Python distributions (free for all uses), but a whole new ecosystem for education has grown around the Jupyter Notebook, including auto-graded notebooks, code visualization, and more. For science and engineering, the Python ecosystem offers a full suite of libraries for array computing (NumPy), symbolic computation (SymPy), plotting (Matplotlib), powerful mathematics including linear algebra (SciPy), and high-performance computing (Numba).

Format of the modules and learning experience

Each module will focus on an applied problem that needs to be solved with computing as a tool and is aligned with topics presented in other engineering courses. The modules will be self-paced, online, richly documented, and focused on mastery for application. Each will contain:

- richly-formatted and interactive documentation (text, equations, and figures)
- step-by-step executable tutorial code
- sub-goal labeled instructions (Margulieux et al., 2012).

Students will be led to run the tutorial code step-by-step, interact with it, extend or modify it, and obtain a solution to an achievable problem. The expected output of practice tasks will be provided so students get immediate feedback. The end- and sub-goals of each module will be contextualized around the expository applied problem, driving the students towards a clear objective throughout.

The skills covered in each module will be identified in collaboration with the course instructors, and through learner analysis of second- and third-year engineering students, such that the application of

each skill is clear and the link to other coursework is obvious. "Dig deeper" sections will include references to textbooks or links to online material where the students can learn more, especially aimed at the high-achieving students who might complete the modules more quickly.

Success in the modules will be a requirement for completion: students will work on each module until they achieve mastery of the problem and concepts it uses. Learning for mastery is backed by extensive research evidence as having not only an overall positive effect on all students, but also reducing the achievement gap (Guskey, 2007). Various innovative competency-based programs are emerging in which the tenet is mastery, not grades. One example is College for America, whose presentation literature reads: "[the] primary form of assessment is completion of a task" (College for America, 2013). Programming skills are ideally assessed in this manner and each module will have an integrated whole-task problem-based assessment to verify student mastery.

The courses below are initial targets for the learning modules. We will interact with the faculty teaching these courses to develop appropriate problem-driven contexts. We have also identified a list of fundamental mathematical and scientific computing concepts to be included in the introductory modules. Below are course subjects with associated computational methods:

- Linear Algebra: matrix-vector manipulation, systems of equations, linear transformations
- Dynamics: solutions of ODEs, motion capture, animations
- Statistics and Experimental Methods: importing/filtering data, curve fitting,regression, FFT
- Linear System Dynamics: Laplace transforms, state-space solutions
- Electromechanical Control System Design: Plotting root locus, Bode, and Nyquist plots

Example content

A typical dynamics course covers the dynamics of particles, rigid bodies, and interconnected systems. Learning objectives include identifying the forces and moments acting on a body and describing its motion under these conditions. Students solve equations of motion analytically when possible and compare to computational solutions. A module supporting this course can focus on projectile motion. Students taking this class at GW build ping-pong ball launchers and use motion capture and a pressure plate to track the trajectory and time of impact. A computational module would develop competencies in the following areas that supplement this project but could be completed asynchronously:

1. Symbolic solution of systems of ordinary differential equations (ODEs)
2. Numerical solution of systems of ODEs using built-in solvers with Runge-Kutta methods
3. Plotting analytical and numerical solutions side-by-side and quantitatively analyzing errors
4. Motion capture from video recordings (tracking the center of the ball)
5. Estimating velocity and acceleration from position data (or vice versa)
6. Creating animations to visualize the motion of bodies or systems of bodies, tracking the motion of specific points on a body such as the center of mass

For more concrete examples of the anticipated format, style and scope of the lessons, see the previous collections by the Barba group: AeroPython, CFD Python and Numerical-MOOC (see References).

Instructional Design

The design of the modules will include rigorous design processes for achieving alignment of module objectives, assessments, and instructional strategies. Our design foundation will be the integrated results of learner and instructional analyses (Jonassen, Hannum, Tessmer, 1989), leading to detailed performance objectives for each module (Mager, 1962). The objectives will link to appropriate within-module student assessments, as well as performance assessments linked to engineering courses. A combination of problem-based learning, scenarios, and direct instruction (Hirumi, 2014) may be used across modules to achieve different learning objectives. We will apply systematic formative evaluations to verify that the materials are not only technically functional, but also effective in achieving student outcomes. The complete instructional design and development processes, and related documentation, will also be made publicly available for other researchers who may wish to alter aspects of one or more modules without reducing the efficacy of a module's (or the series') design.

Our training program falls within the blended-learning category, with additional features: (i) the learning is asynchronous, self-paced; (ii) is supported by face-to-face interactions with learning assistants, i.e., peers; and, (iii) the online materials are designed for interactivity (versus passive learning, e.g., video lectures). This mix allows for the modules to attend to three types of interactions that support learning; instructor-student, student-student, and student-content (Moore, 1989).

The materials will attend to principles of adult learning theory (Knowles, Holton, Swanson, 2005), embedding problem-based learning activities in each module (with instructional assessments and strategies). The content of modules will focus on being relevant and applicable to undergraduate engineering students by aligning module activities with engineering challenges they encounter in their discipline courses. In the STEM-Lab activities, LAs will facilitate high-quality group work and peer interaction, seeding expert problem solving.

Cloud infrastructure for online, dynamic, personal learning

The cloud infrastructure created for this training program includes two components: one public-facing and open access; the other, private for the local student cohort.

The public cloud infrastructure will essentially allow us to offer the training program as a MOOC. We will customize and deploy (on AWS) our own instance of the learning platform created by MIT, Harvard and Berkeley (with contributions from Stanford) for the EdX corporation. The Open edX platform was released as open-source software in June 2013, and the PI already has experience with the self-hosted instance used for her MOOC in Fall 2014, with the support of the specialist consultants at IBL Studios. That instance is still live (and users continue to register for her course, surpassing 7,000), but it uses a very old version (the third release, from April 2015, code-named Cypress). Open edX has evolved considerably in the last two years (with two more named releases, Dogwood and Eucalyptus), and our external consultants must do a new installation from source (no upgrade is possible). We also plan to enable special functionality, not available in the core code base: the capability of issuing open badges for completed modules. In addition, we will undertake the software development of integrating nbgrader (described below) with the grade-book in Open edX.

This work will be directed by the PI but executed by the external consultants (IBL Studios).

The local cloud infrastructure will consist of a JupyterHub server. JupyterHub is a hosted Python data-analysis environment (more details below). It facilitates learning to program in Python because students can run code on a remote server, without installing any software on their local computer. They instead run code on the JupyterHub server through a web browser, reducing the barriers to entry. It also provides a hosting solution where student-generated code (e.g., formative or summative assessment of the modules) can be submitted, and an instructor or learning assistant can run the exact code that the student submitted, on the cloud, or the notebook can be auto-graded.

State-of-the-art educational technology

Interactive computing with Jupyter Notebooks:

The Jupyter Notebook interweaves formatted text with equations (rendered with LaTeX code and MathJax), plots, images and video, and executable code. It is a web-based application, running in a user's web browser and connecting to an execution kernel that is run either locally or in the cloud. Notebooks can be shared online as web pages via the Jupyter Notebook Viewer (nbviewer). They can also be shared by letting users download the notebook document (a text file of the source code) via a public repository like GitHub. Users can then run the notebook code locally, interactively.

Jupyter Notebooks are an extremely versatile tool: equally useful for experimental computations, interactive exercises for students or for presenting polished results. UC Berkeley's pioneering Foundations of Data Science course is delivered via Jupyter Notebooks (Adikhari and deNero, 2016). Other educational offerings include all of the code examples and problems from Downey (2016): "Think DSP: Digital Signal Processing in Python" and the recently published "Python Data Science Handbook" of VanderPlas (2016), written entirely in Jupyter Notebooks and made available for free on GitHub. Jupyter is also being used professionally in science, so students learn with a tool that is immediately applicable in the workplace.The LIGO Scientific Collaboration (2016) released the signal processing analysis of the groundbreaking gravitational-wave detection as a Jupyter Notebook. Companies like Bloomberg are using them for commercial data science, and even contributing to the continued development of Jupyter (Tech and Bloomberg, 2016). And IBM has now integrated Jupyter in their IBM Analytics for Apache Spark cloud data and analytic service (Braida, 2016).

Jupyter Hub:

The software required to run Jupyter Notebooks is free, cross-platform and open-source. Installation takes minutes. However, even a relatively straightforward install presents a level of friction that might turn off new users. To help new users get started with no installation friction, institutions can deploy JupyterHub, a multi-user notebook server that can be run locally or on cloud services like Amazon AWS. End users need only a web browser and an internet connection to use notebooks remotely. Instructors can be sure that every student in a classroom, workgroup or research group has the same software environment with all the required packages for the task at hand. The interface on a JupyterHub server is identical to that of a Jupyter Notebook server run locally, and skills learned on a cloud-based JupyterHub translate seamlessly to the same task on a user's personal computer.

nbgrader:
Just as JupyterHub provides a central entry point to new users to access a known set of software packages, instructors also benefit from extra services provided by JupyterHub extensions. **nbgrader** (Hamrick, 2015) is a free and open-source extension for JupyterHub that allows instructors to automatically distribute, collect, grade and return notebook-based exercises. Students can also "fetch" assignments on their own time, allowing for an easy and autonomous way to track student progress.

nbtutor:
nbtutor (Page, 2016) is a new learning tool for Jupyter Notebooks. When invoked, nbtutor walks a student through code they have written and shows a visualization of how the compute environment evolves as each line of code is executed. nbtutor serves as a valuable supplement to in-class learning assistants. Even though LAs may be available for support in-person at the STEM lab, students working on their own sometimes get stuck trying to understand their code: nbtutor helps a student discover how a code evolves as each line is executed, and reason about the program's logic.

Open edX:
The Open edX platform is the only last-generation, full-featured, open-source platform for scalable online learning. It powers the edX.org MOOC portal, with more than 5 million users and hundreds of courses, it is used by Stanford Online, and by national MOOC sites in France, China, the Middle East, Japan, Spain and elsewhere. (Amigot, 2015) The George Washington University was the second major US university (after Stanford) to deploy Open edX for a successful (independent) MOOC starting August 2014 (Barba, 2015). The GW instance of Open edX was customized, installed and supported by IBL Studios, the technical partners for this project also. It is the first (and as far as we know, only) Open edX site offering open badges (Barba and Amigot, 2014). More than 7,000 users are registered.

Iterative improvement

We will apply the latest research on e-learning and problem-based learning in designing the materials. We will also conduct systematic learner analyses with focus groups to feed into the improvement process, and work with GW instructional designers to formatively evaluate modules throughout the design and development phases. Prof. Watkins will conduct evaluation, for a new iteration of improvement taking into account student progress and areas of difficulty with the modules.

The modules will be developed openly, and hosted on a GitHub repository. PI Barba has wide reach within informal professional circles of computational scientists. We are confident of having feedback from a wider network of both educators and software professionals during the development of the material, and we may even pick up volunteer contributors along the way. The open process of development on the web will contribute to the iterative improvement of the modules.

Student support

The training differs with a traditional computing class in that an instructor does not hold lectures, and that students are not fully synchronized in their progress through the material. All students can advance at their own pace, but the course instructors intervene when students fall behind or

experience special circumstances. They may also impose partial deadlines for completion of graded assignments. A necessary component of such competency-based learning is strong student support. The format for support in this project will be via online Q&A and walk-in tutoring opportunities (office hours and scheduled tutorials). For the online Q&A, we will use the discussion forum on the Open edX platform. (We will also use the forum posts to feedback into the process of iterative improvement of the modules.) The STEM Lab at Gelman Library will be the home of the face-to-face activities, with walk-in help offered similar to a university writing center, and special events hosted to kindle a local community, similar to "maker spaces" (e.g., activities involving programming for robot games, home automation, internet of things, etc.).

The funded graduate student will assist the PIs to train the LAs and the TAs of the served engineering courses. The LAs will be trained in the computing modules and in ways to support students in applying the programming tools in assignments. LAs also receive training via the GW LA Program. They read literature on how people learn, practice assessing and supporting learning, and analyze data from their classrooms during training. Whereas the funding for LAs is not included in this proposal and will be sourced with the existing LA program, the PI has met with the director of the program, Prof. Tiffany Sikorsky, who commits to collaborating in this project (see letter of collaboration).

Tracking student progress

The progress of participating students will be tracked, and a plan of intervention will be put in place when any student falls behind. An ideal platform for the delivery of the modules would also collect learning analytics. We do not include this aspect in the current proposal, but will track the participating students manually (with support of the learning assistants), and using the gradebook in the Open edX learning platform.

4. Short- and long-term impacts

The project will offer a case study of integration of computing in the curriculum, which will be disseminated widely. The local cohort of students will see immediate benefit in their ability to apply computing to their learning and future careers. Online followers of the effort will be able to not only use the training materials, but also to participate in discussions and complete formative assessment in the public learning platform.

Long-term, we will forge a community to share computational learning objects and best practices. Our software contributions to integrate elements of the Jupyter ecosystem in the Open edX platform will benefit others who want to use these technologies in their instruction (the PI has had direct email communication with edX CEO Anant Agarwal that proves such interest exists).

5. Access and adoption

The PI's MOOC on (graduate) numerical methods has attracted more than 7,000 registered users on the Open edX platform, while the GitHub repository with the course materials has more than 1,000

forks. Clearly there is wide interest in this type of training, and the PI has been able to reach a large audience. For this project, we will build on this with a revamped online platform, publishing all materials openly on GitHub, and disseminating at conferences and via social media.

The greatest challenge for adoption is securing buy-in from other engineering faculty to require or encourage the new materials in their courses. On this issue, co-PI Wickenheiser will serve as test subject himself, and provide the point of view of a faculty member who is not an expert in the scientific Python ecosystem. As a regular Matlab user, he will help identify similarities and differences with that environment, so that we may anticipate switch-over friction points. To entice other faculty to adopt our training program as part of their courses, we will offer a graded assignment that can easily replace existing homework and a lecture-replacement activity. In other words, we save them some work. In addition, we will hold a **"Jupyter for profs"** seminar at the beginning of each semester.

6. Partnerships for collective impact

The PI has commitment from colleagues at other universities (see letters of collaboration) who will trial the materials, offer feedback, potentially modify or contribute peer review, and help build a community around the project. The collaborators have links with the PI as follows. Prof. LeDoux was a co-participant with Barba at the NAE Frontiers of Engineering Education Symposium in 2012, and in 2013 as a co-panelist on online learning. He also co-edited with Barba a recent issue of Advances in Engineering Education dedicated to flipped classrooms (Barba, Kaw, LeDoux, 2016). Prof. Niemeyer is co-editor with Barba in the Journal of Open Source Software, and the Journal of Open Engineering, and a member of the SciPy community. Prof. Huff was Program Co-Chair of SciPy 2014, where Barba keynoted about her educational program—notably declaring Jupyter Notebooks a "killer app" for education (@KatyHuff tweet 7:36 AM - 8 Jul 2014). She was also a BIDS scholar and shared workspace with Barba during her sabbatical in Berkeley, where they discussed computing education often.

The project also will benefit from informal advisors from industry. Currently, the PI is working to revise an existing course in engineering computing. In this work—a preliminary and complementary effort to this project—she has secured the following advisors: Joerg Gablonsky, The Boeing Company: Chair, Enterprise HPC Council; and, Rick Arthur, Director of Advanced Computing at GE Global Research. The advisors will help craft authentic, industry relevant learning outcomes. They will also advise on the expectations for engineering graduates to be computationally skilled, comment on our ideas to integrate computing in the curriculum, and help align our curriculum to industry needs. As we build our relationship with the advisors, we expect to continue to benefit from their input moving into this project. We also will work with them (and other industry contacts) to identify internship opportunities for students that rely on their computing skill.

7. Scalability and sustainability

A previous open online learning initiative of the PI already scaled to thousands of users. The scalability challenge faced in this project is more at the institutional level, where we need a critical mass to make the training program infrastructural in the curriculum. Locally, we will promote the training program

among students via events at the "STEM Lab" that are social, fun, maker-inspired. The experience of the UC Berkeley Foundations in Data Science program shows that students are aware of their need for computational skills, and flock to a good training program (Barba was exposed to this program as a visiting scholar at the Berkeley Institute of Data Science in Spring 2016). Once a local community is thriving, the next level of scalability is attained by supporting our outside collaborators to transfer the experience. Sustainability will be achieved when other instructors start building derived works and new materials using the same methods. We rely on the practices and ethic of the open-source world to support this effort. Previous experience with the PI's Numerical-MOOC is a good example, where the materials were adopted for courses in the UK and Belgium, they were used for a study group in Canada (Miller, 2015) and they have been forked on GitHub by more than 1,000 people.

8. Recruitment and evaluation

We will work with faculty teaching engineering courses to adopt a computational module and require its completion as a graded piece of their course. Local faculty will be enticed with an offer to replace a lecture and provide the graded work with no effort on their part. With this approach, we don't need to recruit students directly for the training program, locally—we will call for volunteers to participate in the evaluation. (Participants will complete informed consent forms prior to participation.)

Implementation monitoring and evaluation:
During the design and development of the modules, we will apply a combination of 4-level training evaluation (Kirkpatrick, 1994) and the Technology Acceptance model, developed specifically for the study of computer-technology acceptance. The 4-level training evaluation component will assess learner satisfaction, learning, utilization, and anticipated value, combining both learner and engineering course-instructor assessments. The first two levels, satisfaction and learning, will be assessed through surveys and quizzes at the end of each module.

Technology Acceptance proposes that acceptance of a technological innovation is determined by two judgements about the technology: perceived usefulness and perceived ease of use. This model is well-documented for studying the acceptance of educational technology by both students and faculty (Teo, 2011). We will use it to assess learner perspectives on the third (utility) and fourth (value) levels of the 4-level training model, through a short pre/post student survey for each module.

Magana et al. (2013) applied the Technology Acceptance model to evaluate a curricular innovation to introduce computing in materials science and engineering program, consisting of a new computing course, plus embedded computational modules in other courses. They focused their evaluation on students' acceptance of computing as a tool for their continued studies and future careers. Following them, we will use a pre/post assessment with both Likert-survey and open-ended questions, intended to measure predictors of future behavior in relation to use of computing in their studies and careers. Questions will address perceived ability (to design an algorithm, to visualize data, etc.), utility, and intention to use in future coursework, projects, and in their careers (sample questions below). We will obtain a composite score in each of these three categories. Students will also be given a survey at the end of the semester asking them to evaluate how well the modules prepared them to do any other programming work required in the supported courses. Some of the questions in this survey will ask

students to identify particularly difficult areas and how they would be better supported by the modules (for iterative improvement of the material).

The Technology Acceptance model is an indirect measure: by interrogating students' perception of ability and utility of computing, we anticipate future behavior in how students might use computing in their studies and careers. Students' perception of ability is an important measure, and in the case of programming, it is particularly useful. If students are confident of their ability to use computing to solve problems, they are more likely to attempt it, and thus continue to get better at it.

Example Technology Acceptance survey questions:
Likert-scale questions of the Technology Acceptance model for computing training from Magana, Falk & Reese (2013): "I have the ability to design an algorithm; I have the ability to write a computer program; I have the ability to use a computer to solve a set of linear equations; I have the ability to visualize data using a computer; I have the ability to numerically solve an initial-value problem; I have the ability to implement a numerical model based on a simple differential equation; I feel computation will be useful in my studies; I feel computation will be useful in my career; I intend to seek courses that will allow me to increase my knowledge about computation; I intend to use computation in my future career."

Data related to student satisfaction, learning, and technology acceptance will be analyzed after each semester in order to make improvements to the design and implementation of the modules. Data from multiple semesters will be analyzed to identify trends and assess the value of the modules.

Instructor Feedback:
At the end of each semester, we will interview the instructors of the disciplinary courses to assess their perceptions about general student performance and preparedness to apply computing for problem solving. Both the student and instructor feedback will be used to improve later versions of the modules, to create new modules to support the most difficult concepts, or to provide references to additional back-up materials.

Outcome Evaluation:
Working in collaboration with industry partners, we will develop an authentic workplace scenario that requires the application of skills developed upon completion of the all modules. When half of the participants in the research have completed all modules then the scenario will be provided to all participants, including those that have not completed all of the modules. The participants' abilities will to apply the module content to solving the workplace challenge will be assessed, and these scores will be analyzed in relation to the number of modules the participants have completed. It is anticipated that there will be a significant correlation between completing the modules and successfully completing the workplace challenge.

The second year engineering students, including those who are participating in the evaluation, will also be taking a traditional programming course. Hence the workplace scenario evaluation should provide an indicator of the modules meeting a desired outcome of the project: improving the ability of participants to apply computational thinking and programming to real-world workplace challenges.

9. Broader Impacts

The training program will form computationally skilled engineers who are better prepared to enter the workforce competitively, or if moving on to graduate studies, are ready to use computing effectively as a research tool. The language (Python) and the teaching environment (Jupyter) are immediately applicable in research and industry. By integrating computing in the curriculum, the training achieves the goals without increasing time to degree. It will serve as a template for other institutions to adopt, and provide materials that are publicly available to re-use, modify and build from (shared under permissive licenses). We recognize that adoption and impact depend on building a community. The community building for this project is already started, through the PI's previous efforts (supported by her NSF CAREER award), and is reflected by the letters of collaboration. It will expand online through the PI's network (which includes more than 7,000 users of her graduate-level MOOC).

The impact will be reinforced from applying the collaborative ethos of open source. Using open-source tools and teaching learners about the open-source world has the added value of showing effective coordination of teams via online platforms. Open-source projects are able to create value from collective work thanks to a culture of commitment and transparency. Our training will emulate this culture, building an online community and enticing local students with activities at the STEM lab.

The GW engineering class of 2020 is 40% female, twice the national average, and the GW LA program is 60% female. We commit to a target of at least 50% women among the LAs trained for this project. Outward-looking, a context-based presentation is known to benefit female students.

10. Experience of the investigators

PI Barba developed a set of lessons for her Computational Fluid Dynamics (CFD) course (Boston University, 2010–2013) that led students to code a solution of the governing equations of fluid dynamics (Navier-Stokes). The lessons were proven in the classroom to effectively take students with almost no programming experience to the point where they could write a classic (but non-trivial) solution of CFD, in about four weeks of classes. Based on this, she created a module called "CFD Python" (Barba, 2013a; Barba, 2013b), consisting of 12 Jupyter Notebooks with full-text material and Python code. In a two-day intensive course, she saw some students completing the same work that stretched over four weeks in a classroom setting, while most students completed over ¾ of the work (at each student's pace).This experience, though anecdotal, gave Barba the confidence to envision a self-paced learning environment for computational engineering, supported by interactive notebooks.

Barba has a long track record of teaching innovations, starting with screencasts to support a lecture-based course, whiteboard capture and digital inking, then publishing open courseware since 2010. She has full collections on iTunes U for courses in Fluid Mechanics (junior), CFD (senior and graduate), and Bio-Aerial Locomotion (freshman). She was the top content provider of the BU iTunes U channel (as shown by the system analytics). In 2012 she edited and moved the CFD videos to YouTube, where they have passed 500,000 views. She began using the "flipped classroom" method in 2012 and has written and given talks about this pedagogical approach. She has been consistently engaged in

educational innovation, takes a scholarly approach that includes inquiry into learning theory and teaching practices, and openly disseminates results from her experiences and educational products.

Prof. Wickenheiser has been deeply involved in curricular development, especially in computational methods and computer-aided design, since his appointment in 2010. He served on the department's undergraduate curriculum committee for two terms, he spearheaded a brand-new Robotics option and revamped the capstone design sequence. He developed new programming and CAD content for eight undergraduate courses. He also co-authored a series of MATLAB programming labs, covering image processing and manipulation, for the freshman Introduction to Engineering seminar. He has mentored members of the AIAA student branch in developing annual MATLAB workshops for undergraduates and robotics workshops for Northern Virginia high schoolers.

Prof. Watkins has taught, written about, and conducted research on various topics of instructional design and online learning since 1998. He co-authored the "E-learning Companion: A student's guide to online success," with over 150,000 copies in print and now in its 4th edition. It is one of the few books focusing on preparing learners for online courses. In 2005 he authored 75 E-learning Activities to help instructors find creative ways to make online courses more engaging. He has authored or edited seven other books and more than 90 articles on e-learning, needs assessment, or related topics. He is the developer of the WeShareScience.com online platform for sharing research. With his unique experiences as an online instructor, instructional designer, and evaluator, he works to assess the ability of e-learning to meet the requirements of learning in varying contexts. He often consults with the World Bank on e-learning development, instructional design, and program monitoring and evaluation.

11. Results from Prior NSF funding

Lorena A. Barba

NSF award # OCI-1149784, CAREER: Scalable Algorithms for Extreme Computing on Heterogeneous Hardware, with Applications in Fluids and Biology; PI: Lorena Barba.
Total Award Amount: $550,627 | **Period of performance**: 3/01/2012–2/28/2017
Summary: Involves new algebraic applications of the fast multipole method (FMM) in elliptic solvers and preconditioners, and in fast matrix-vector products in iterative methods for solution of algebraic equations. The applications are in fluid dynamics, and boundary integral solutions of bioelectrostatics and Stokes flow problems. The educational program includes open educational resources and flipped classroom approaches for computational fluid dynamics and numerical methods. **Products:** Comput. Phys. Comm., 184(3):445–455 (2013); Comput. & Fluids, 80:17–27 (2013); SIAM News, 46(6):1 (July 2013); Comput. Phys. Comm., 185(3):720–729 (2014); J. Chem. Phys., 143:124709 (2015); Comput. Phys. Comm., 202:23-32 (2016); J. Open Source Software, 1:43 (2016).

Ryan Watkins and Adam Wickenheiser

The co-PIs have no prior NSF-funded research efforts to report.

References Cited

Adams, Joel C. **Hot Job Market for Computer Science Graduates**. Communications of the ACM, Vol. 57, No. 1, p. 19 (Jan. 2014).

Adhikari, Ani and deNero, John (2016) **Computational and Inferentinal Thinking: The Foundations of Data Science**, https://www.inferentialthinking.com (open textbook under CC-BY). GitHub repository https://github.com/data-8/textbook

Amigot, Michael (2015) **What Makes the Open edX Platform Unique? See These Cases**, Open edX blog https://open.edx.org/blog/what-makes-open-edx-platform-unique-see-these-cases

Barba, L. A. (2013) **CFD Python: 12 Steps to Navier-Stokes**. Blog post http://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/

Barba, L. A., et al. (2013) **CFD Python**. GitHub repository (Jupyter Notebook source) on https://github.com/barbagroup/CFDPython

Barba, L. A. (2014) **Announcing AeroPython**! Blog post http://lorenabarba.com/blog/announcing-aeropython/

Barba, L. A. and Mesnard, Olivier (2014) **AeroPython**. Figshare https://dx.doi.org/10.6084/m9.figshare.1004727.v3 and GitHub repository (Jupyter Notebook source) https://github.com/barbagroup/AeroPython

Barba, L. A. (2014) **Announcing "Practical Numerical Methods with Python" MOOC**. Blog post http://lorenabarba.com/news/announcing-practical-numerical-methods-with-python-mooc/

Barba, L. A., et al. (2014a) **Numerical-MOOC.** GitHub repository (Jupyter Notebook source) https://github.com/numerical-mooc/numerical-mooc

Barba, L. A., et al. (2014b) **Practical Numerical Methods with Python**. Online course platform http://openedx.seas.gwu.edu/courses/GW/MAE6286/2014_fall/about

Barba, L. A. and Amigot, Michael (2015) **Preview: Open edX extension to use digital badges in a MOOC**, figshare https://dx.doi.org/10.6084/m9.figshare.1243674.v2

Barba, L. A. (2015) **What's Open edX?** Class Central MOOC Report https://www.class-central.com/report/whats-open-edx/

Barba, L. A. (Mar. 2016) **Computational Thinking: I do not think it means what you think it means.** Blog post https://medium.com/@lorenaabarba/computational-thinking-i-do-not-think-it-means-what-you-think-it-means-6d39e854fa90#.tlvnimk9h

Barba, L. A., Kaw, Autar and LeDoux, Joseph (2016) **Guest Editorial: Flipped Classrooms in STEM**, http://advances.asee.org/publication/guest-editorial-flipped-classrooms-in-stem/

Braida, Andrea (2016) **Accelerating data science with Jupyter Notebooks and Apache Spark,** IBM BlueMix blog https://www.ibm.com/blogs/bluemix/2016/03/accelerating-data-science-with-jupyter-notebooks-and-spark/

College for America (2013), presentation slides in PDF (see slide 12) http://collegeforamerica.org/site_images/Reduced_College_for_America_Public_Version.4.16.13.pdf

CRA, Computing Research Association. **Computing Degrees and Enrollment Trends, 2011–2012 Taulbee Survey**, May 2013.

Downey, Allen B. (2016) **Think DSP: Digital Signal Processing in Python**, O'Reilly Media (1st ed.) ISBN-10: 1491938455. GitHub repository (Jupyter Notebook source) https://github.com/AllenDowney/ThinkDSP

Hafner, Katie (2012). **Giving Women the Access Code**, The New York Times, April 2, 2012 http://www.nytimes.com/2012/04/03/science/giving-women-the-access-code.html

Hamrick, Jessica (2015) **nbgrader**, GitHub repository https://github.com/jupyter/nbgrader

Guskey, Thomas R. (2007). **Closing Achievement Gaps: Revisiting Benjamin S. Bloom's "Learning for Mastery,"** Journal of Advanced Academics, vol. 19(1):8–31, Sage Publications.

Guzdial, M. (2010). **Why is it so hard to learn to program?** In Andy Oram and Greg Wilson, editors, Making Software: What Really Works and Why We Believe It, Chapter 7, pp. 111–124. O'Reilly Media, 2010.

@GWEngineering tweet (official Twitter handle of GW SEAS), 11:01 AM - 23 Nov 2016: Help make SEAS the #1 school in the US for graduating women w/ science & engineering degrees http://bit.ly/2f013QY #WOMENinSTEM https://twitter.com/gwengineering/status/801440574020734976

GW Today (Oct. 30, 2016), **STEM lab to host tutoring, workshops for students** (quoting Barba) http://www.gwhatchet.com/2016/10/30/library-lab-to-host-tutoring-workshops-for-stem-students/

GW Undergraduate Learning Assistants (LA) Program, http://tsikorski.wixsite.com/gwula

Hirumi, A (2014). **Grounded Instructional Strategies**. Retrieved Jan. 15 2017 from http://sitios.itesm.mx/va/dide/docs_internos/docs_enc/hirumi/h01strategies.pdf

Jonassen, D. H., Hannum, W., and Tessmer, M. 1989. **Handbook of Task Analysis Procedures** Westport, CT: Praeger Publishers.

@KatyHuff tweet, 7:36 AM - 8 Jul 2014: #scipy2014 keynote @LorenaABarba says "IPython Notebooks are the Killer App for teaching (science and engineering)." https://twitter.com/katyhuff/status/486519116451176448

Knowles, M. S., Holton, E. F., Swanson, R. A. (2005). **The adult learner: The definitive classic in adult education and human resource development**. Boston: Taylor & Francis Ltd.

Larson, Selena (2013). **Schools Aren't Teaching Kids To Code; Here's Who Is Filling The Gap,** blog post on ReadWrite, October 18, 2013. http://readwrite.com/2013/10/18/kids-learn-code-programming

LIGO Scientific Collaboration and Virgo Collaboration (2016) **Signal processing with GW150914 open data**, https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html

Magana, Alejandra J., Michael L. Falk and Michael J. Reese, Jr. (2013). **Introducing Discipline-Based Computing in Undergraduate Engineering Education**. ACM Trans. Comput. Educ. 9(4), Article 39 (April 2013), 22 pages.

Magana, Alejandra J., Michael L. Falk, Mike Reese, Camilo Vieira (2013). **Materials Science Students' Perceptions and Usage Intentions of Computation**, 120th ASEE Annual Conference & Exposition, June 23–26, 2013. http://www.asee.org/public/conferences/20/papers/7450/view

Mager, R. (1962). **Preparing instructional objectives** Atlanta: Center for Effective Performance.

Mannila, Linda and Michael de Raadt (2006). **An objective comparison of languages for teaching introductory programming**. In Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006 (Baltic Sea '06). ACM, New York, NY, USA, 32-37. http://doi.acm.org/10.1145/1315803.1315811

Mannila, Linda, Mia Peltomäki, Tapio Salakoski (2006). **What About a Simple Language? Analyzing the Difficulties in Learning to Program.** Computer Science Education 16(3): 211–228.

Margulieux, Lauren E., Mark Guzdial, and Richard Catrambone (2012). **Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications**. In Proceedings of the ninth annual international conference on International computing education research (ICER '12). ACM, New York, NY, USA, 71-78. http://doi.acm.org/10.1145/2361276.2361291

McCracken, M. V. Almstrum, et al. (2001). **A multi-national, multi-institutional study of assessment of programming skills of first-year CS students**. SIGCSE Bulletin, Vol. 33(4), pp. 125–180.

Meyer, Bertrand (1993) **Towards an object-oriented curriculum.** In Proceedings of the 11th International TOOLS Conference, pp 585–594, Prentice Hall.

Meyer, Bertrand (2003). **The Outside-In method of teaching introductory programming**, In Perspective of System Informatics, Proceedings of the 5th Andrei Ershov Memorial Conference, Akademgorodok, Novosibirsk, 9-12 July 2003, eds. Manfred Broy and Alexandr Zamulin, Lecture Notes in Computer Science 2890, Springer-Verlag, pp. 66-78.

Miller, Jonah (2015) **Course Review: Practical Numerical Methods with Python by George Washington University**, Class Central MOOC Report https://www.class-central.com/report/gwu-numerical-mooc-review/

MIT Media Lab (2016a) **In Memory: Seymour Papert**. https://www.media.mit.edu/people/in-memory/papert

MIT Media Lab (2016b) **Learning from Seymour Papert**, video on YouTube https://youtu.be/Pvgef9ABDUc?t=43m10s

Moore, M.G. (1989) **Three types of interaction**, American Journal of Distance Education/3/2, 1-6.

National Research Council of the National Academies (2011). **Report of a Workshop on the Pedagogical Aspects of Computational Thinking**, Committee for the Workshops on Computational Thinking. http://www.nap.edu/catalog.php?record_id=13170

NPR (August 5, 2016) **Remembering A Thinker Who Thought About Thinking.** http://www.npr.org/sections/ed/2016/08/05/488669276/remembering-a-thinker-who-thought-about-thinking

Page, Logan (2016) **nbtutor**. GitHub repository https://github.com/lgpage/nbtutor

Papert, Seymour (1980) **Mindstorms: Children, Computers, and Powerful Ideas.** Basic Books, Inc. New York, NY; 1980. ISBN:0–465–04627–4

Papert, Seymour (1996), **An Exploration in the Space of Mathematics Educations**, International Journal of Computers for Mathematical Learning, Vol. 1, No. 1, pp. 95–123.

Porter, Leo, Mark Guzdial, Charlie McDowell and Beth Simon (2013). **Success in Introductory Programming: What Works?** Communications of the ACM, Vol. 56, No. 8, p. 34.

President's Council of Advisors on Science and Technology (PCAST). **Designing a Digital Future: Federally funded research and development in networking and information technology** (Dec. 2010)

Soloway, E., J. Bonar, et al. (1983). **Cognitive strategies and looping constructs: An empirical study**. Communications of the ACM, Vol. 26, No. 11, pp. 853–860.

Tech at Bloomberg (July 2016) **Inside the Collaboration That Built the Open Source JupyterLab Project**, https://www.techatbloomberg.com/blog/inside-the-collaboration-that-built-the-open-source-jup

yterlab-project/

Teo, Timothy (ed.). **Technology Acceptance in Education**, Sense Publishers (2011) https://www.sensepublishers.com/media/1035-technology-acceptance-in-education.pdf

The Journal of Open Engineering, http://www.tjoe.org

The Journal of Open Source Software, http://joss.theoj.org

U.S. Dept. of Homeland Security, **Characteristics of H-1B Specialty Occupation Workers Fiscal Year 2009 Annual Report**, April 2010.

University of Colorado Boulder, **Colorado LA Model** https://laprogram.colorado.edu/node/8

VanderPlas, Jake (2016) Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media (1st ed.), ISBN:978-1-4919-1205-8. GitHub repository (Jupyter Notebook source) https://github.com/jakevdp/PythonDataScienceHandbook