Justin Pearson's CCUT Portfolio

Certificate for College and University Teaching
University of California, Santa Barbara

November 08, 2015

CONTENTS

CHAPTER

ONE

SIGNATURE PAGE

UCSB **CERTIFICATE IN COLLEGE AND UNIVERSITY TEACHING**
University of California Santa Barbara

SIGNATURE PAGE

Please type or print all of the required items on this form, gather the appropriate signatures and insert the pages into the front of your CCUT Portfolio. Submit one paper copy of your portfolio to Lisa Berry, 1130 Kerr Hall, and an electronic version to lisa@id.ucsb.edu. Your portfolio should be submitted as early as possible and no later than the quarter BEFORE you expect to graduate.*

Name: Justin Pearson Email: justin.pearson@gmail.com

Department: Electrical & Computer Engineering Phone: 925-200-6844

Name of Department Faculty Graduate Advisor: Nadir Dagli, dagli@ece.ucsb.edu

REQUIREMENT 1

- Serve at least two quarters as a Teaching Assistant or Associate at UCSB

Course #	Employment Title	Quarter/Year
ECE 147C	Teaching Assistant	Spring 2014
Engineering 3	Teaching Assistant	Summer Session B, 2014

- Complete all TA Training activities required by your department

Departmental Faculty Graduate Advisor's signature verifying completion of the above two activities:

JL
JOAO HESPANHA FOR
Signature (Nadir Dagli) 8-31-15 Date

- Attend the day-long campus-wide Teaching Assistant Orientation
- Receive a classroom videotaping and consultation

TA Development Program Coordinator's signature verifying completion of the above two activities:

Signature (Kim or Lisa) 9/16/15 Date

REQUIREMENT 2

- Complete EITHER a CCUT-approved course in pedagogy OR attend a CCUT-approved program in teaching (SCWriP Summer Institute, Two quarters as video consultant fro TADP, Lead TA Institute, Summer Teaching Institute for Associates, or a Writing Program TAship. See website for details)

Course # and Name OR Teaching Program Name	Quarter/Year
CSEP's School of Scientific Thought	Jan - Mar 2015

Instructor's signature verifying completion of Requirement 2 (or attach a copy of transcript, certificate, etc.):

Signature (Wendy Ibsen) 3/9/15 Date

*If you will graduate in Summer Quarter, your portfolio must be submitted by the 5th week of Spring Quarter.

Submit as first page of CCUT Teaching Portfolio to: Lisa Berry, Co-Chair of CCUT, Instructional Consultation, 1130 Kerr Hall.

UCSB CERTIFICATE IN COLLEGE AND UNIVERSITY TEACHING
University of California Santa Barbara

REQUIREMENT 3 (Complete one of the following)
CHECK ONE

- ☐ Option 3A: **DISCUSSION** of your own experiences using or implementing instructional technologies to enhance student learning. This 8-10 page discussion (double-spaced, 12 pt. font) must be supported by published research.
- ☐ Option 3B: **RESEARCH REVIEW** of existing research on effective instructional strategies using instructional technologies (~10 pages, double-spaced, 12-point font). Practical examples from candidate's own teaching experience and observations should be integrated into the research review.
- ☐ Option 3C: **INSTRUCTIONAL TECHNOLOGY PROJECT** demonstrating creative and substantive development of instructional materials utilizing computer-based or multi-media to enhance student learning. The project must be a significant aspect of a course.
- ☒ Option 3D: Completion of an **APPROVED COURSE** for Req. #3. Approved courses include: INT 223 A, B or C; ED 256, or ED 253D.

Course # ED 253D: Blended Learning — Quarter/Year Spring 2014

KDeBacco (Kim) — Instructor's Signature — Date 09/16/15

REQUIREMENT 4
CHECK ALL

- ☒ Taught course as an instructor of record with mentoring support of a faculty member.
- ☒ Included all necessary documentation in the CCUT Portfolio:
  - A letter from mentor describing the nature and frequency of the mentoring (A Summer Teaching Institute for Associates (STIA) certificate may be offered in lieu of a mentor's letter).
  - ESCI or other course ratings
  - Open-ended student evaluations with discussion of your strengths and weaknesses

REQUIREMENT 5
I have submitted all of the materials required for completion of the CCUT Teaching Portfolio and attest to their accuracy.

Student's Signature — Date 2015-09-16

Release of CCUT Portfolio Contents

"I ☒ DO ☐ DO NOT release my CCUT portfolio for use as an example to other CCUT applicants if the Faculty Advisory Board so chooses." (List any sections you wish to have omitted in a separate page.) **Omit all Appendices except Appendix 3.**

"Further, I give permission for the above indicated parts of my proposal to be presented as examples in the following forms."
☐ As a hard copy ☒ On the CCUT website

"I ☒ DO ☐ DO NOT want my name attached to my portfolio if it is used as an example in the above forms."

Student's Signature — Date 2015-09-16

Submit as first page of CCUT Teaching Portfolio to: Lisa Berry, Co-Chair of CCUT, Instructional Consultation, 1130 Kerr Hall.

CHAPTER

TWO

MY TEACHING PHILOSOPHY

My teaching philosophy is encapsulated in three themes:

1. The instructor must provide a purpose and a path for the student.

Learning takes effort, and students need to be convinced that the effort is worth their time. Therefore, when designing a lesson, I first ask myself "why will my students want to listen to me?" I strive to make the material engaging and relevant for my students. For example, for an introductory programming course last Summer, I demonstrated how students could write programs to monitor online apartment listings and send notification emails when new listings appeared. This got their attention because many of them were in the midst of house-hunting for Fall quarter. In a lesson about linear algebra, I showed students how they could use nutrition data provided by Jamba Juice to reverse-engineer smoothie recipes. To introduce Graph Theory, we discussed the "6 Degrees to Kevin Bacon" project, which links actors based on which movies they've acted in together. My students engaged with the material when they could relate to their own lives and interests, and I enjoyed getting them excited about learning.

Once my students are eager to learn about a topic, learning can begin. I arrange the topics like building blocks, ensuring that each new concept builds upon its predecessor without requiring outside knowledge. For example: "Last time we talked about vectors and strings," I'd say in my programming class. "We found they were useful for storing numeric data or character data. But soon we'll encounter problems where it's natural to store both types of data in a single object. Consider a bank account; it has a balance — stored as a number — and the owner's name — stored as a string. It would be nice to link the number and the string together in a single datatype. Today we will learn about a new datatype called a 'struct' that does just that." In this way, the students progress through the concepts along a logical path, each step building on and reinforcing previous ones.

2. Struggle is a vital component of learning, but must be carefully modulated.

Students learn best when they can practice new material; grappling with new skills is how we build proficiency in them. However, not all struggling results in the same quality of learning, as any attendee of a late-night cramming session will attest. Therefore I strive to present my students with many opportunities for low-stakes practice, well before any homework due-date or exam. For example, I pepper my lectures with in-class exercises and questions for the students. Sometimes the questions have short, unambiguous answers: "Okay, I've drawn a triangle for the brick's velocity, and we're interested in this side, this other side, and this angle. Sine, cosine, or tangent?" For more involved problems, a think-pair-share or group discussion activity may be appropriate: "Based on our discussion of function syntax, write a function to solve the quadratic equation". While the students work, I circulate around the room to check in on the students. After bringing the class back together, the students volunteer their approaches and we discuss it. I sometimes cold-call the students using a deck of cards, so that the students know they're on the hook. This also serves to get a uniform sampling of the students' status, instead of polling just the talkative ones. But if a student can't answer, I don't push it: I put their card aside for a later question. My main goal is to give my students ample opportunities to struggle with the material in a low-risk way. I believe that one of the chief roles of an educator is to foster a balance of healthy, productive struggling in students.

3. The instructor must be able to empathize with the student's struggle.

As educators, we must modulate our students' struggle carefully: left unchecked, struggle leads to frustration and resentment. To strike this balance, an educator needs empathy. I use a variety of methods to keep my fingers on the

pulse of my students' welfare. With index cards, I solicit weekly feedback from my students — "What's going well? What is challenging for you?" — and adapt my class accordingly. For example, contrary to my expectations, my students reported that they wanted even more in-class exercises than we already had. I was happy to oblige, and the students felt validated and empowered. Also through feedback cards, I learned that students felt I lectured a little too slowly, so I picked up the pace. Feedback cards have been invaluable for learning what my students need.

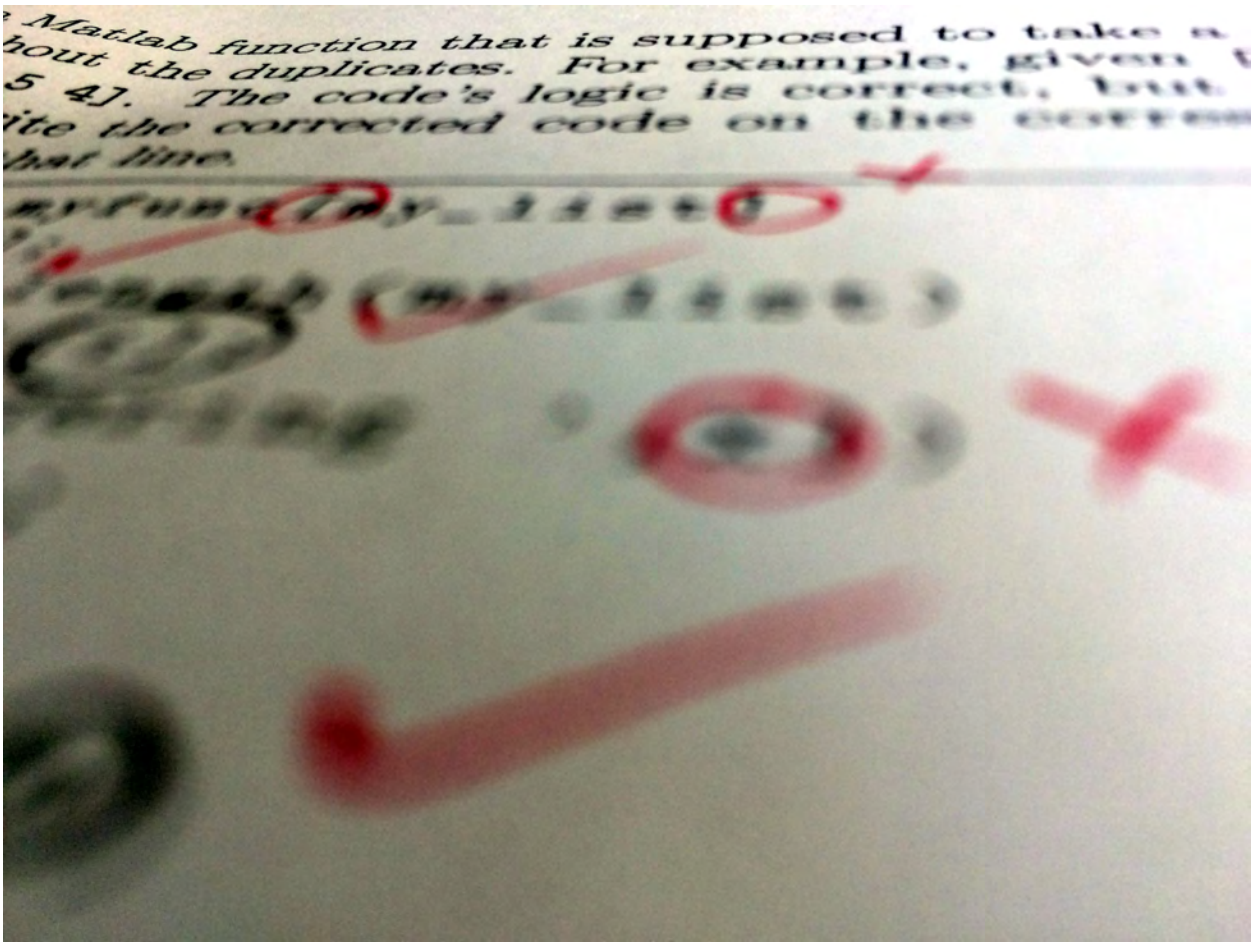
Empathy takes many forms beyond simply asking what students need. When writing my lessons, I pretend I'm seeing the material for the first time and I ask myself, "what questions do I have at this point in the lecture?" When lecturing, I keep the tone informal and conversational, because I remember being afraid to ask dumb questions of my brilliant, formal professors. When helping a frustrated student, I take care to be sympathetic and supportive because I remember how it feels to be stumped on a problem the night before it's due.

A pessimistic professor once told us on the first day of class: "Students are the only group of people who are happiest when they aren't getting their money's worth." Who can blame them? Without a good teacher to provide motivation, guidance, and empathy, school is just one big pointless, frustrating effort — and we are all creatures trying to minimize effort. By adhering to these educational values, I hope to help students embrace the struggle of learning instead of shying away from it.

CHAPTER

THREE

REQUIREMENT 1: SERVE AS A TEACHING ASSISTANT



ENGR 3 (Intro to Programming) midterm, Summer 2014

3.1 Executive Summary

I served as a Teaching Assistant for ENGR 3: *Introduction to Programming* and ECE 147C: *Control Systems Design Project*. The former introduces Freshmen engineers to the Matlab programming software that they will use throughout their college careers, whereas the latter is a lab-based upper-division course that teaches sophisticated engineering design methods to juniors and seniors. The student populations between these two courses are on opposite ends of the undergraduate engineering spectrum, and require different kinds of TA support. The TA Training event provided valuable general-purpose teaching tips, and the one-on-one TA Video Consultation offered me personalized feedback: talk slower and increase student participation. I took this feedback to heart and received very favorable ESCI teaching evaluations. The students appreciate my energy, enthusiasm, and empathy.

3.2 TA Training and Consultation

The UCSB-wide TA Orientation and one-on-one TA Consultation provided some very useful teaching tips. I worked hard as a TA and Instructor of Record to integrate them into my teaching. For example:

- I tried hard to **learn the students' names**, as a show of respect and validation. My students indicated on open-ended feedback cards that they appreciated this gesture.
- I peppered my lectures with short in-class **think-pair-share exercises** for the students. Initially I thought students would find them cumbersome, but students reported via feedback cards that they wanted even more in-class exercises. I was pleased to find that students liked the practice and ensuing discussion.
- I was careful to use **non-judgmental language** when responding to a students' question or incorrect answer. A student called me "patient and kind" in his open-ended feedback.
- I tried to **face the audience** when writing on the board and writing code at my computer. I think that making eye-contact encouraged students to participate.
- My in-class exercises had **clear start/stop signals**: "Okay, so take 3 minutes to try this out, and I'll call out when we're done. Go for it!"
- I tried to **ask clear questions**, e.g., instead of "what should I do next?", I would ask "what is the first line of a function declaration?"
- I was careful to **not answer my own questions**. Sometimes this required waiting several uncomfortable seconds for students to volunteer an answer, but as TA Orientation speaker Scott Dirkse says, "That is the sound of them learning."
- I was careful to **ask the question, then call on the student**. Otherwise students might zone out once they know they're off the hook.
- I wrote their names on cards in order to randomly choose students. I was afraid that only the hotshot students were answering my questions, and I wanted a way to **uniformly sample the students' understanding**.
- I took care to **not block the board**.
- I distributed **feedback cards** to the students to get their input on how the course was going.

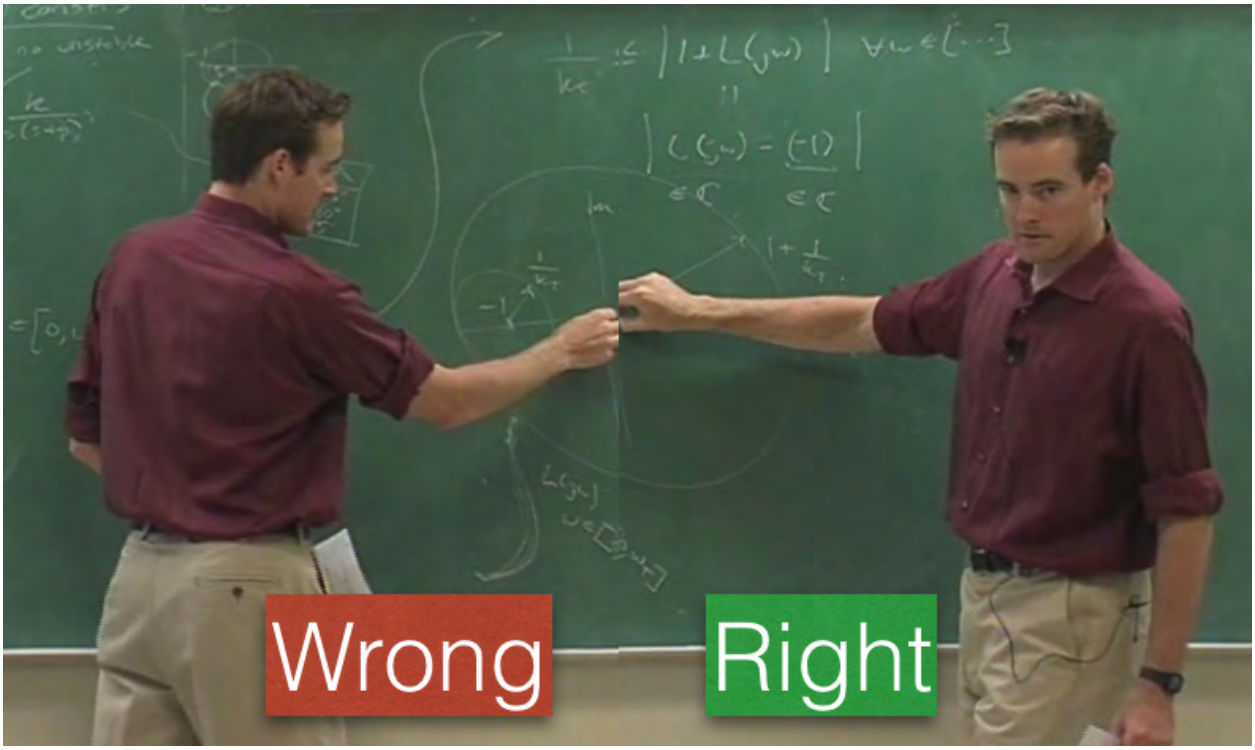
I found it natural to employ many of these tips, since many of them reflect the themes in my teaching philosophy. For example, learning a student's name and using non-judgmental language are examples of an *empathetic teacher*. Conducting appropriate in-class exercises and soliciting questions from students fall in the category of *make the students struggle a little*. And simple things like using clear start/stop signals and not blocking the board exemplify a *well-structured lesson*.

While the TA Orientation provided general-purpose teaching advice, my one-on-one TA Video Consultation offered concrete, personalized feedback. For example, UCSB Instructional Developer Kim DeBacco pointed out that my

teaching style was pretty fast and a little frenetic; my enthusiasm came across as nervous energy. This was transformative because it highlighted how my teaching style was at odds with my teaching philosophy's emphasis on *struggle* and *structure*: enthusiastic lecturing has its place, but in excess it detracts from the structure of the lesson and it deprives the students of their own opportunity to struggle.

As a result of the TA Video Consultation, I tweaked my teaching style to incorporate more student participation. Kim taught me some techniques for slowing down, breathing, and positioning myself with respect to the board, and she suggested preparing more questions to ask my students. I employed those techniques a few months later, when TAing ENGR 3 in Summer 2014, and a year later, when teaching ENGR 3 as Instructor of Record. Instead of charging along for 90 minutes, excitedly talking and gesturing, I would instead ask a question or assign a short in-class exercise for the students. This increased student participation and interest, and students responded well to it in their weekly feedback cards. Moreover, I still retained my enthusiasm, and students remarked in their feedback cards that they appreciated my passion and energy.

Kim DeBacco's writeup from my video consultation appears in *Appendix 2*.



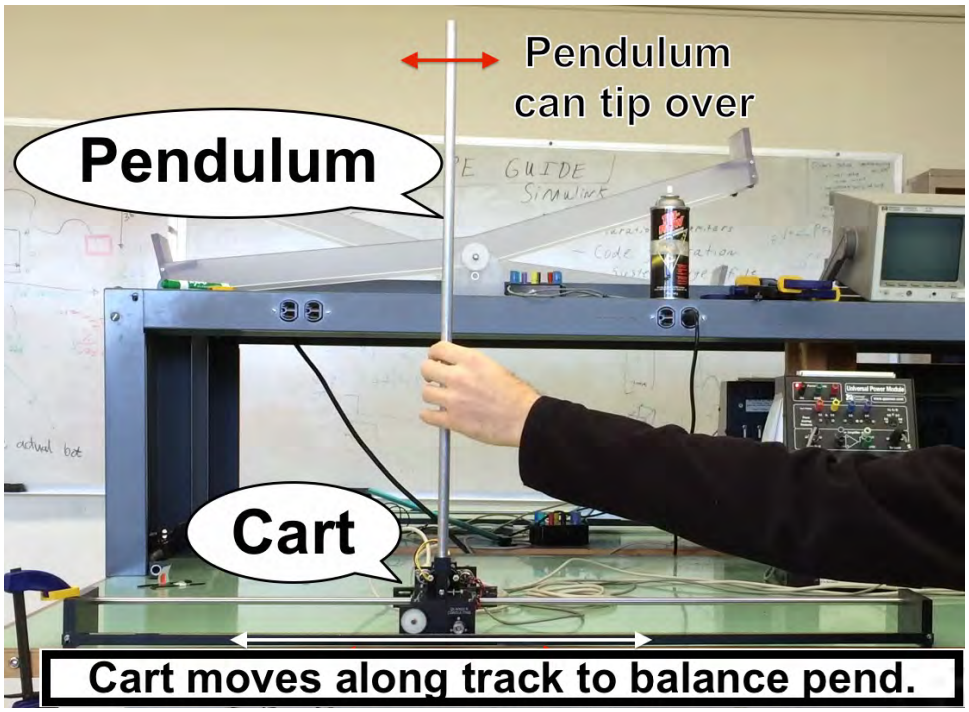
Frames from my guest-lecture of ECE 147C — it's better to face your audience!

3.3 Course Descriptions

3.3.1 ECE 147C: Control Systems Design Project

Date:	Spring 2014
Course:	ECE147C / ME106A
Description:	Control Systems Design Project / Advanced Mechanical Engineering Laboratory
Enrollment:	9 ECE, 1 ME, 3 UCSB Extension (Foreign Exchange students)

ECE 147C is a lab-based class where junior- and senior-level engineering students apply the principles of *System Identification* and *Control Theory* to write a computer program to balance an inverted pendulum:



The cart and pendulum setup used in ECE 147C. (video)

For most students, ECE 147C offers their first chance to apply control theory to real hardware. This is exciting, but there are ample opportunities for frustration: faulty lab equipment; friction in the track; nonlinearities in the cart model; software bugs. Empathy — one of the themes in my teaching philosophy — was my key asset to aid the students' learning in this environment. Often, the students' code wouldn't work and the pendulum would tip over and slam into the desk. Frustrated, the students would call for help, and I found that having an open, inquisitive, non-judgmental attitude helped set the stage for a fruitful debugging environment. It also helped that I had taken ECE 147C as a Mechanical Engineering undergraduate at UCSB in 2006, so I had suffered through the class and I knew how frustrating it could be.

3.3.2 ENGR 3: Introduction to Programming

Date:	Summer 2014
Course:	ENGR 3
Description:	Introduction to Programming
Enrollment:	44

ENGR 3 is an introductory programming course for Freshmen engineers. Using the Matlab programming environment, students learn the basic datatypes and control structures that underlie any program. About halfway through the course, students are introduced to more sophisticated mathematical objects like differential equations, and the students learn to solve them using built-in Matlab routines. Summer Sessions instructor Dr. Lina Kim taught ENGR 3, and I TA'd it with another TA, Michael Nip. Lina wrote the lectures and exams, and Mike and I wrote and graded the homework assignments, and ran the discussion sections.

One challenge with ENGR 3 was the students' wide spread in programming experience. Some students were in the Freshman Summer Start Program (FSSP), meaning that they had started college a quarter early in order to get a jump-start on classes. Other students were retaking ENGR 3 because they failed it the first time. And still other students were not engineering students at all, but needed ENGR 3 to satisfy their department's programming requirement. Moreover, some FSSP students took programming classes in high school, whereas others had never programmed at all. How can a teacher engage and guide students with such varied backgrounds? How should an instructor create different "learning pathways" for these different groups? As Jaime Escalante from "Stand and Deliver" says, "how can I reach these kids?"

Class Level	******* Majors *******		Objectives
15 FR	13 EE	1 PHYS	15 BA
12 SO	8 CHEME	1 ENVST	28 BS
13 JR	7 ME	1 PRECM	1 MA
3 SR	4 PRCHM	1 ECON	MS
1 MA	4 UNDEC	ENGR	PHD
P1/P2	2 PRECO		Other
Other	2 BIOCS		

Enrollment data for ENGR 3, Summer 2014. Students are spread across many years and many majors.

We approached this challenge in several ways. First, we let students choose which discussion sections to attend, regardless of which one they were enrolled in. Mike's discussion section was more advanced and theoretical, and mine focused on emphasizing the core class concepts. This allowed experienced students to explore extensions to the material with Mike, while novice programmers could reinforce their understanding in my section. It turned out that most students opted to attend my discussion section.

Next, I learned my students' names and asked them many questions. This kept my discussion section informal and increased participation, so I was able to ascertain my students' well-being and focus the discussion where it was needed.

Also, Lina, Mike, and I used feedback cards judiciously throughout the course: Every Wednesday, Lina would distribute index cards and ask students to write what was going well and what needed improvement. We scheduled my office hours immediately after Lina's lecture, so after class she would deliver the cards to me and I could react immediately to student concerns. Feedback cards provided a key mechanism to stay in touch with the students' welfare and tune our office hours and discussion sections to suit the students' needs. From a thematic standpoint, feedback cards helped me empathize with my students. For example, motivated by student feedback, I tweaked my discussion sections to involve problems more closely related to the homework. This was the first of many lessons in empathy: from the students' perspective it's natural that grades are the highest priority.

I took this feedback to heart. Instead of highlighting interesting applications in discussion section, I folded those applications into the homework assignments. The students seemed to enjoy them; one student even wrote a comment in his code thanking me for devising such fun homework problems!

I was interested to observe that different types of help and empathy were needed between ECE 147C and in ENGR 3. Seniors in ECE 147C were in the home stretch of their college careers and had somewhat "checked out". They felt more aloof and would only ask for help as a last resort. I found it fruitful to have them explain their code to me as a peer, rather than have me dictate the correct code to them. For them, I played the role of a fellow engineer, helping them debug a common problem. On the other hand, my ENGR 3 students needed much more hand-holding. Taking their first tentative steps in college, they needed more guidance and reassurance. For the ENGR 3 students, I played the role of a mentor and guide.

3.4 ESCI Teaching Evaluations

3.4.1 ECE 147C

The ECE course evaluations are much shorter than the ENGR ones, so I received less feedback from ECE 147C. Five of my 11 students filled out ESCI evaluations for ECE 147C. This low turn-out was a bit disappointing but not unexpected since most of the class was seniors who were struggling to finish their senior projects at the time. My ESCI evaluations from ECE 147C were very positive:

Please rate your TA on the following scale.

Poor TA	0%
Marginal TA	0%
Adequate TA	0%
Good TA	0%
Best TA this quarter	40%
Best TA ever	60%

In their open-ended feedback, the students considered me helpful and a fair grader, and they appreciated the extra lab time. Their ESCI open-ended feedback was sparse, but they posted two reviews on www.ratemyprofessor.com:

- Justin is a great TA! He was very enthusiastic about the class, and seems truly interested in the material. He gave me some really good explanations, that was helpful in understanding the material. I have no doubt that he will make a great teacher/employee in the future.
- Justin was very helpful and I think that he actually cared about helping us on the material. He is knowledgeable about the subject matter and very punctual and lab session in office hours. He consistently emailed us to keep us updated and gives us tips on the homework or lab reports. He is the ideal TA.

I felt encouraged by this feedback. It seems I was able to provide a good learning path for my students and empathize with their struggle.

3.4.2 ENGR 3

Twenty of the 24 students in my discussion section submitted ESCI reviews. For ENGR 3, the ESCI reviews contained 20 questions and an open-ended response question. Histograms of my performance appear in *Appendix 4*. From these data I observe:

What's going well

- The students found me enthusiastic, clear, knowledgeable, and approachable.
- They liked my lecture slides.

What needs improvement

- Some students felt I moved too fast, while others felt I was a bit repetitive. This could be due to the large spread in abilities in ENGR 3, but it's certainly possible I sped through some parts and harped on others. I probably

should have asked for more in-class feedback of the form "thumbs up if you'd like me to go faster, down if you want slower." I employed this technique frequently a year later, when I taught ENGR 3 as Instructor of Record.

- Some students felt that discussion section only contributed moderately to the course. I imagine that those students wished my discussion sections had been more directly applicable to the homework; I took care to not give homework answers in section, opting instead for developing the week's concepts with my own examples. In hindsight, I realize that students care primarily about their grades, so if a discussion section doesn't clearly relate to improving their grade, they may tune out. In the future, I will try to find a suitable balance between giving away answers and holding their attention.
- Some students felt I only somewhat conveyed the usefulness of the subject matter. In an introductory class, it is sometimes difficult to motivate the very basic material with cool examples. But it is important to try to make the material as interesting as possible. After all, one of my teaching themes is: *The instructor must provide a purpose and a path.* So I will try to find more stimulating examples. (Note: A year later, when I taught ENGR 3 as instructor of record, this backfired because I made the homework problems too stimulating (too hard)! So there is a balance.)

There were five open-ended responses in the ESCI surveys:

- Best TA I ever had. Very helpful and makes you want to learn the material. Hire him as a professor.
- Justin is a great TA. Very easy to talk to you. Very friendly and enthusiastic.
- Enthusiastic and really helpful. Thoroughly explains material when asked.
- Sometimes he takes a while to explain something but I always understand by the end which is more important so I didn't bother me too terribly.
- Amazing TA. Always make sure his students understand the material. Also learned everybody's names.

I appreciate this feedback. It identified ways I can be more empathetic — e.g., by focusing my teaching on what the students care about — and indicated that I succeeded in my efforts to connect with the students — e.g., by learning their names and offering friendly explanations.

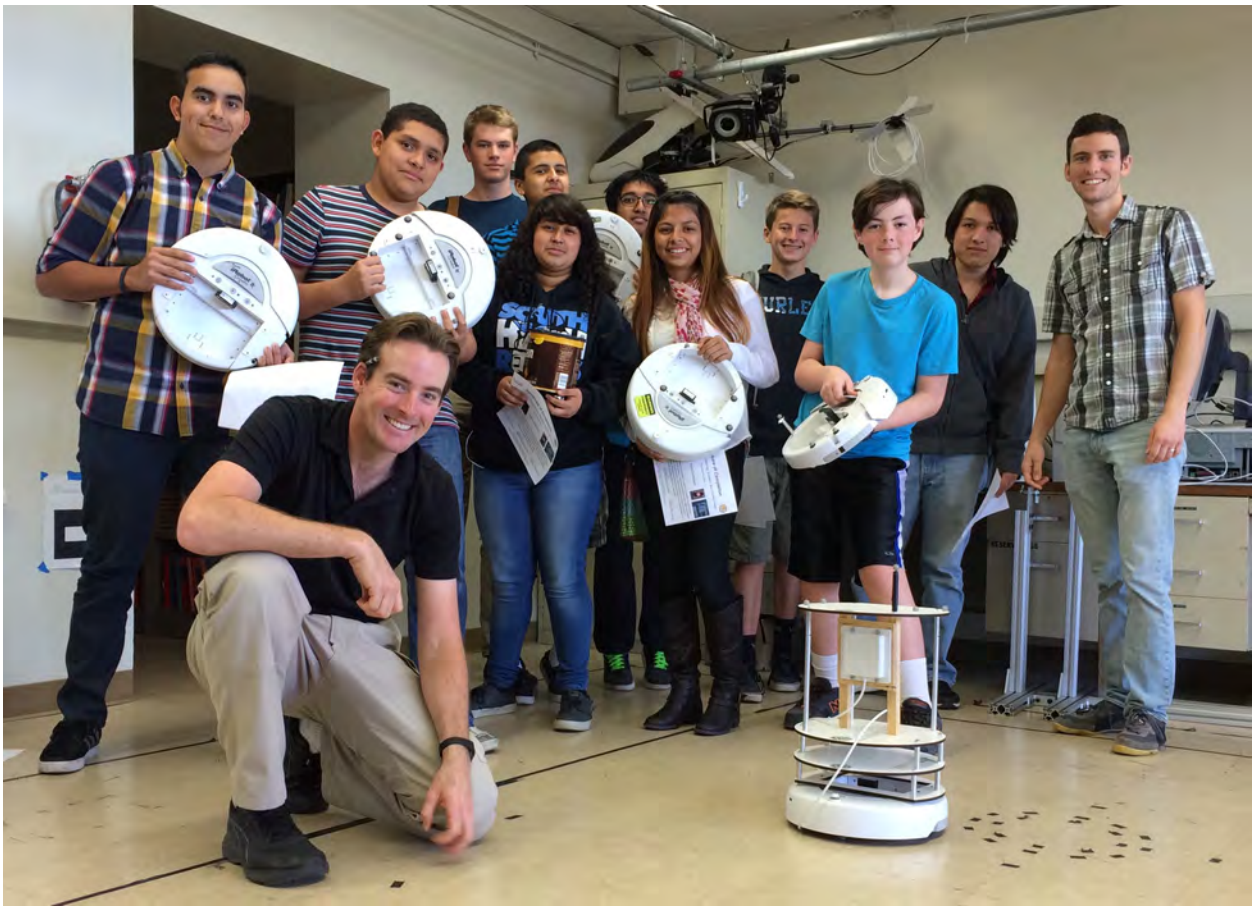
3.5 Conclusion

I really enjoyed the variety of TAing a lower-division course and an upper-division one. I was interested to explore how my two different sets of students required different kinds of empathy. As discussed, I adopted different teaching roles — peer, mentor, guide — to best suit my students. This helped me grow as an educator: to be flexible and adaptable for my students' needs. I leveraged these lessons when teaching my own course in Summer 2015, ENGR 3: *Introduction to Programming*.

CHAPTER

FOUR

REQUIREMENT 2: RUN A TEACHING PROGRAM



School of Scientific Thought, March 2015

4.1 Executive Summary

My colleague and I designed a course to teach robotics, control theory, and programming to high-school students, and taught it through the *School of Scientific Thought* outreach program. Over the course of five Saturdays, the students explored these concepts by programming Roomba robots. Each session involved a short lecture about a technical topic, followed by a programming session in which the students worked in teams. We experimented with several instructional techniques, including gallery walks, think-pair-shares, and feedback cards. Because of the students' wide spread of programming and mathematical abilities, we took care to establish a respectful classroom environment and to select programming activities of appropriate difficulty. Students indicated via an exit survey that they enjoyed the program.

4.2 Description of SST

The School of Scientific Thought (SST) is an outreach program offered twice a year by the Center for Science and Engineering Partnerships at UCSB. Over the course of 5 Saturdays, roughly 70 high-school students come to UCSB to participate in hands-on courses prepared by UCSB graduate students. Each high-school student is assigned to a course based on his or her preferences. The courses vary from year to year, depending on the available graduate students. After each 2-hour course session, the students and mentors from all the courses meet for lunch. This gives the students a chance to talk to the mentors in an informal setting, and ask them questions about college. At the end of the five week program, the students get certificates. The main goal of the SST program is to give high-school students a glimpse into what college is all about, and to get them excited about continuing their education after high-school.

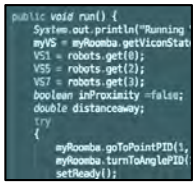
UCSB

Certificate of Completion

Introduction to Programming, Robotics, and Control Theory

This certifies that


Marlene Benitez

has successfully completed a five-week course on computers, programming languages, robotics, and control systems. You programmed ground robots to achieve specific goals with both low-level serial communication and a high-level Java programming environment. These skills will serve you well in your future career in nearly any technical field. Good job!

Justin Pearson, Course Instructor

Wendy Ibsen, Associate Director CSEP

David Copp, Course Instructor

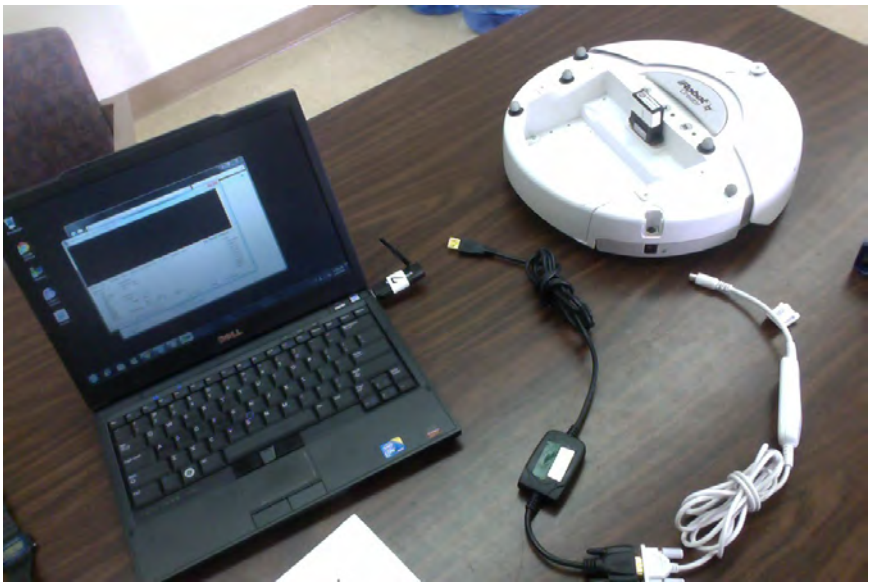
Center for Science and Engineering Partnership
California NanoSystems Institute, University of California, Santa Barbara

Example certificate given to our SST students.

4.3 Course Design and Reflection

In March 2015, my colleague David Copp and I designed the SST course "Introduction to Programming, Robotics, and Control Theory," which was advertised to the high-school students as follows:

> Computers outnumber humans on planet Earth, and they impact almost every aspect of our lives. So it's increasingly valuable to know how they work and how to control them. In this class you will learn how to program computers in order to control robots. You'll work in teams to program a Roomba-like robot to perform exciting tasks based on the robot's locomotion and sensor capabilities. You'll write code to make the robot react to its environment, which forms the basis for the wild world of feedback control. Most importantly, you'll learn to translate high-level goals like "push this hockey puck into the goal" into a language the machine understands. These skills are fundamental to broad fields like Computer Science, Control Theory, and Mobile Robotics. This course provides a heavily hands-on and project-based learning environment, and no previous programming experience is required!



The Roomba robot and the laptop used to program it. We had 5 Roombas, with 3-4 students working on each.

Each of our five sessions consisted of a short lecture followed by a programming segment. Based on advice from UCSB Instructional Developer Lisa Berry, we designed lesson plans to organize each day:

Topic & Time	What the facilitator is doing	What the students are doing	Materials needed
Introduction 10:00 - 10:20	• quick intro • ice-breaker • 'code of conduct' *(emphasize importance of respectful communication)*	pairs talk to each other, then you talk about your partner	main whiteboard, presentation
The ubiquity of computers and robots 10:20 - 10:40	• Prompt for gallery walk: "Where do you see computers and robots?"	gallery walk	big posterboards or whiteboards

A mental model for thinking about control systems
10:40 - 11:00
	• Showing videos
	  ○ 'what do all these have in common?'
	  ○ (*write kids' responses on board in roughly the right bubbles)*
	• talk about 4-bubble flowchart
	• The Roomba fits into this flowchart.
		• Raise hand volunteering info
		• TPS: example of flowchart: a car & a roomba
			main whiteboard

Introduction to the Roomba	• Hand each group a roomba. • Based on their responses to roomba flowchart, point out the hw:	The kids are holding roombas and looking at the hardware, volunteering guesses at what	Roombas

Our lesson plan for the first SST session.

The lectures typically revolved around some aspect of robotics or programming, preparing the students for the day's programming project. The programming projects typically involved tasks like "write a Java program to drive the robot in a square" or "send a binary command to the Roomba to flash its LED." The short lectures were interspersed with gallery walks and think-pair-shares. We kept the lectures short — less than 30 minutes — to maximize the time the students had with the fun, novel robots. The programming sessions were chaotic, with teams driving robots all around the room, talking with each other, and struggling through software bugs. My co-instructor and I spent this time drifting between teams, helping them debug, and offering insight.

My SST co-instructor (right) helps students debug their Roomba robot.

Our main challenge stemmed from the students' wide range of programming and mathematical abilities. When designing our class, we knew we would be teaching Freshmen alongside Seniors, with any range of programming experience in between. In such a situation, it's natural for students to judge each other. In order to prevent an intellectual hierarchy from forming among the students, we took care to establish a respectful classroom environment. Specifically, on the first day, we explained that we knew that some students had taken programming classes whereas others had never programmed before. We established a "code of conduct" by outlining respectful behavior. I illustrated the importance of respect by sharing a story from my experience at an aerospace company, where a hostile work environment played a significant role in the failure of a very expensive product. The story and the code of conduct had the desired effect: students treated each other respectfully and nobody's feelings were hurt. During one group brainstorming activity, I actually heard a student turn to a shy classmate and earnestly ask, "What do YOU think?" I was pleasantly surprised at this demonstration of empathy from a high-school student.

Our students' wide range of abilities also influenced our decision to not have a programming competition. Previous robotics internship mentors in our lab have used an end-of-class programming competition to motivate the students to work. In the case of SST, the drawbacks outweighed the benefits:

- If we grouped the students by skill level, the novice student teams would be unfairly matched against the experienced student teams.
- On the other hand, if we grouped the students diversely, dispersing the experienced programmers among the teams, we feared each team would have their best programmer hog the laptop to maximize their chance of winning.

We let the students choose their own groups. Instead of instituting a competition, we merely presented a goal for each day, like "write a program to drive your robot in a square." It turned out that the robots were sufficiently novel that they held the students interest, so a competition wasn't needed.

In addressing both of these potential pitfalls, the predominant pedagogical theme was *the instructor must empathize with the students' struggle*. In this case, the "struggle" pertains to the pressure students feel when comparing them-

selves to each other in the classroom.

Me lecturing at the beginning of an SST session.

It is worth noting two more consequences of the large spread of student abilities. First, it offered us an opportunity to practice delegating some teaching to the students. Specifically, we found it helpful to encourage the students to explain concepts to each other during the programming segment of the class. Explaining a concept solidifies learning, so it is valuable to both the explainer and the explainee.

4.4 Student Feedback

The students' feedback is presented in *Appendix 6*. The students reported that they really liked the course. They liked the programming sessions more than the lectures, although they reported that they learned a little less in the programming part of the class.

Here are some useful comments we received from the student on improving the course:

- "Have longer classes: 8-10 weeks." This possibly indicates that we were too ambitious in the amount of material we tried to teach. In the future, we might provide the students with fully-functional code that the students can change. This would be easier and more fun for them than learning to write it themselves, although slightly less educational. That would be more aligned with SST's mission, and would provide a more clear learning pathway for the students.
- "Hands-on example by the teachers first." Typically in the lecture portion, we showed slides demonstrating what the students should do, and we gave the students printouts of the slides. Yet students were sometimes unsure of how to get started. They probably would have found it more helpful to follow along with a hands-on demo. However, that introduces the problem of how to proceed if a team gets stuck. Instead of forcing the students to proceed in lock-step, we opted for the more independent and flexible approach of outlining the approach and helping them as needed.
- "Use a programming language other than Java." The Java programming language is a little hard to learn because it is "strongly-typed," meaning that the programmer must declare the type of every variable (string, number, array, etc). Python would be a more attractive alternative, but we have a significant Java infrastructure written already. We are considering non-Java alternatives.

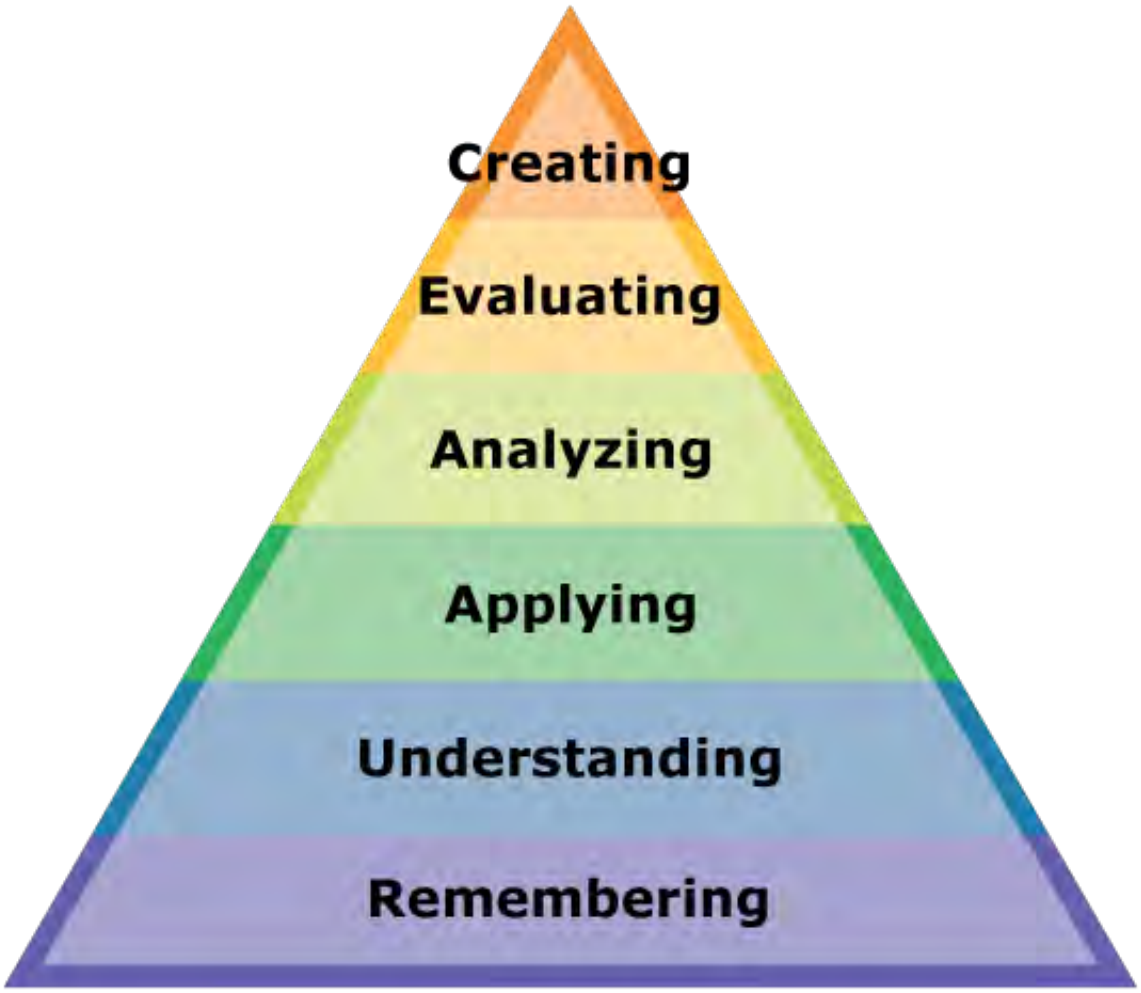
4.5 Conclusion

The SST let me practice teaching high-school students in a low-risk way: They were there to have fun, and I enjoyed thinking of activities they would enjoy. I learned techniques for managing large numbers of students from different backgrounds. I found that establishing a respectful and fair learning environment avoided conflict and promoted an empathetic working environment. In keeping with the importance of struggle in a student's learning, I designed our SST course to encourage our students to struggle a little bit, without overwhelming them. I learned to appreciate the power of having students work in groups as a way to reinforce their learning. I used many of these techniques six months later, when teaching ENGR 3 as Instructor of Record.

CHAPTER

FIVE

REQUIREMENT 3: TECHNOLOGY-SUPPORTED PEDAGOGY



Bloom's Taxonomy of Educational Objectives

5.1 Executive Summary

In Spring 2014 I took the course ED 253D: *Blended Learning*, taught by UCSB Instructional Developer Kim DeBacco. The course explored how to blend online learning technologies with traditional learning techniques. The main deliverable for the course was the design of my own blended course. I prepared a "flipped" course, teaching the foundations of mathematics and logic: I wrote the syllabus, outlined the course's topics, made a class website, wrote online quizzes and sample blog posts, and recorded the first few video lectures. ED 253D was extremely valuable to my future career in education because it taught me relevant pedagogical theories and provided a framework for me to practice them. In terms of my teaching philosophy, ED 253D equipped me to provide a well-structured learning path for my future students.

5.2 My online course: "Skills for Thinking Technically"

For my ED 253D project, I designed a flipped course in foundational notation in mathematics, logic, and technical writing. I was motivated by my observation that engineering students often struggle with new concepts due to a poor grasp of underlying mathematical objects and notation.

This project helped me grow as an educator because it allowed me to practice several pedagogical theories and technologies. Here are some highlights of the course:

5.2.1 Learning Outcomes

In ED 253D we learned that a vital first step when designing a lesson or course is to identify your desired learning outcomes for your students, e.g., "By the end of this course, my students will be able to ...". Good learning outcomes are specific and testable. Writing good learning outcomes helped me structure my ED 253D project, and it helped me organize my lessons in subsequent courses.

Here are some examples of my course's learning outcomes:

- Dissect an argument using formal logic
- Recognize common logical fallacies
- Manipulate common mathematical objects and notation
- Recognize abuse of mathematical notation in technical writing and infer its meaning

5.2.2 "Flipped" Classroom: Synchronous and Asynchronous Activities

With Learning Outcomes in place, I next had to decide what learning activities would support them. Moreover, each learning activity (lecture, homework, project, quiz, discussion section, etc) has its own advantages in the context of a flipped classroom. I found it very useful to classify a learning activity by two criteria:

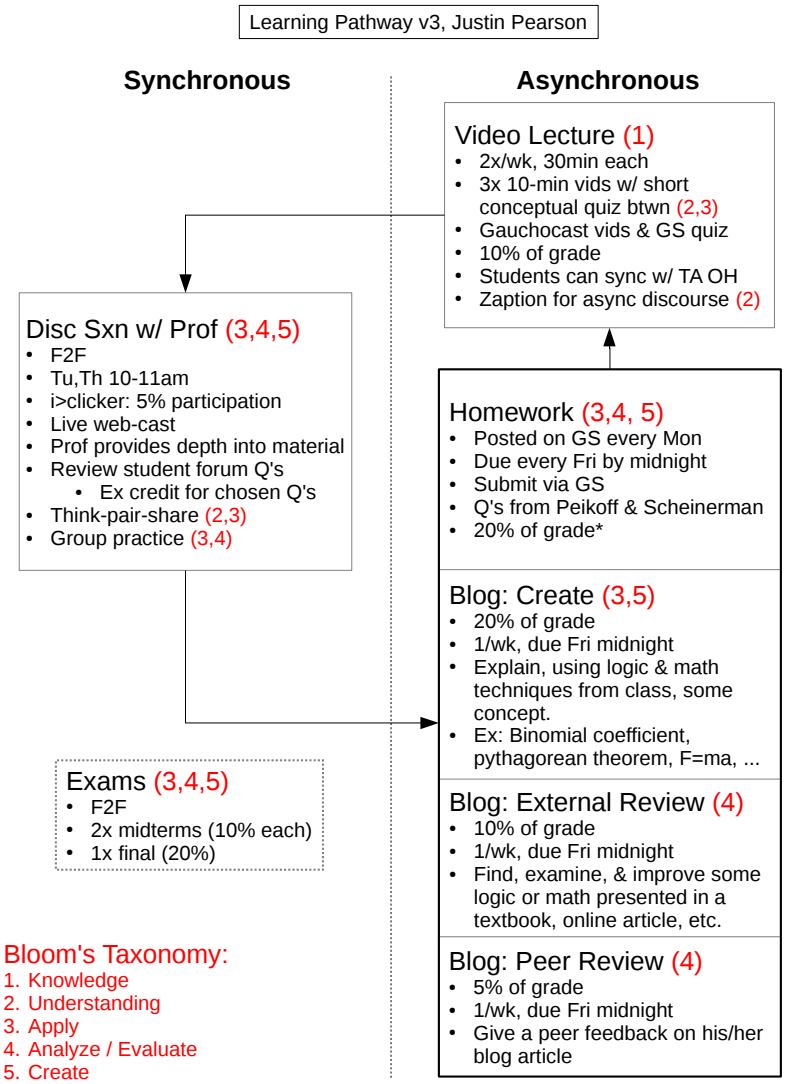
1. *Is it synchronous or asynchronous?* Are the student and the instructor synchronized in time? (Practically speaking, synchronous means the student can ask questions of the instructor in real-time.)
2. *Is it face-to-face or distant?* Are the student and the instructor co-located?

For example, the traditional "broadcast-style" lecture would be a synchronous, face-to-face interaction, whereas a homework assignment or a pre-recorded video lecture would be asynchronous and distant.

These categories capture the trade-offs inherent in various learning activities. For example, asynchronous activities give students freedom to access the activity at will, but do not give them the opportunity to ask questions. Face-to-face interaction helps students stay focused, but requires the students and instructor both be physically present. These two dichotomies helped me weigh the relative strengths of these categories.

5.2.3 Learner Pathway

After classifying several candidate learning activities as *synchronous* vs. *asynchronous* and *face-to-face* vs. *distant*, I decided on the ones I wanted in my class. I then designed a "Learner Pathway" that describes how my students navigate through my course:



The "Learner Pathway" outlines how students navigate my class.

This offered me the opportunity to practice the pedagogical techniques we learned in ED 253D. For example, note the red labels on the activities above. These labels indicate which steps of *Bloom's Taxonomy of Educational Objectives (Appendix 7)* are satisfied by the activity. For example, "Video Lectures" is labeled (1) because the first level of learning in Bloom's Taxonomy is "introduce the new knowledge," and it is through the video lectures that students are introduced to the new material. The short conceptual quizzes interspersed with the video are labeled (2,3) because they help the students understand the concepts (Bloom level 2) and apply the concepts to solve simple problems (Bloom level 3).

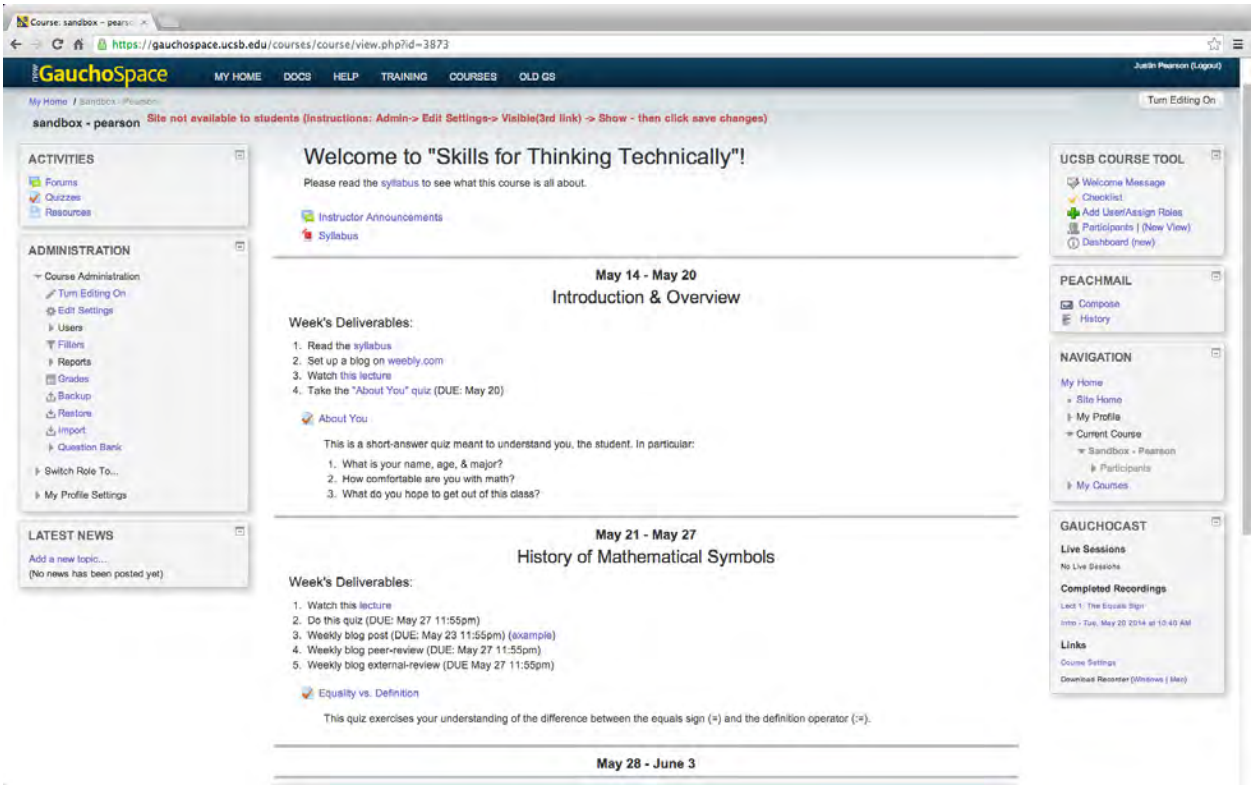
I found Bloom's Taxonomy to be a very useful tool for structuring how students' learning develops through my course.

5.2.4 GauchoSpace course management system

Guest lecturers in ED 253D warned us that students sometimes find online courses nebulous because there is no instructor to remind them what they should be doing and when they should do it. Therefore it was vital for the online portion of my flipped class to present information clearly and concisely.

I decided to use the GauchoSpace course management system (CMS) to host the online portion of my class. GauchoSpace is widely used at UCSB, and is the UCSB-tailored version of the popular Moodle CMS. Learning GauchoSpace wasn't difficult, and it was an important step for me to become a better educator.

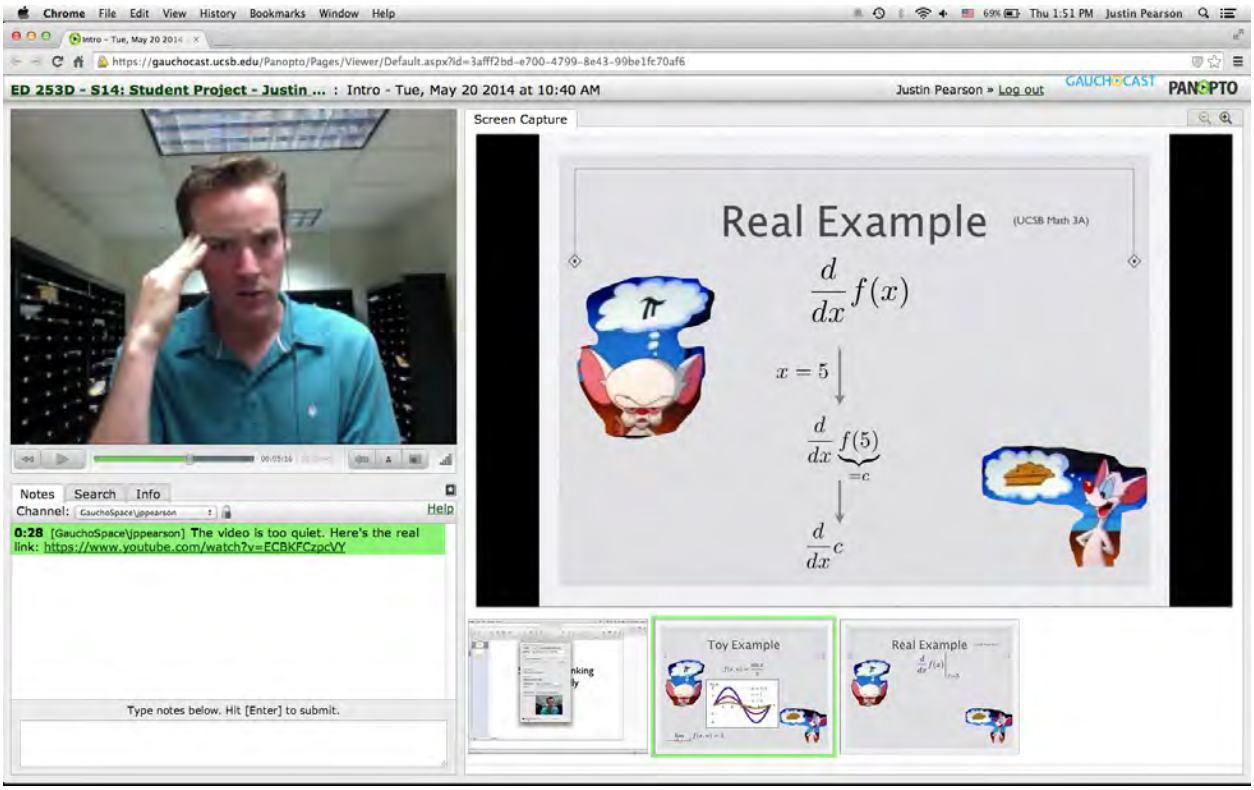
Here is a screenshot of my course's home page on GauchoSpace, including posting the syllabus and an online quiz:



The GauchoSpace online website for my course.

5.2.5 Online video lectures

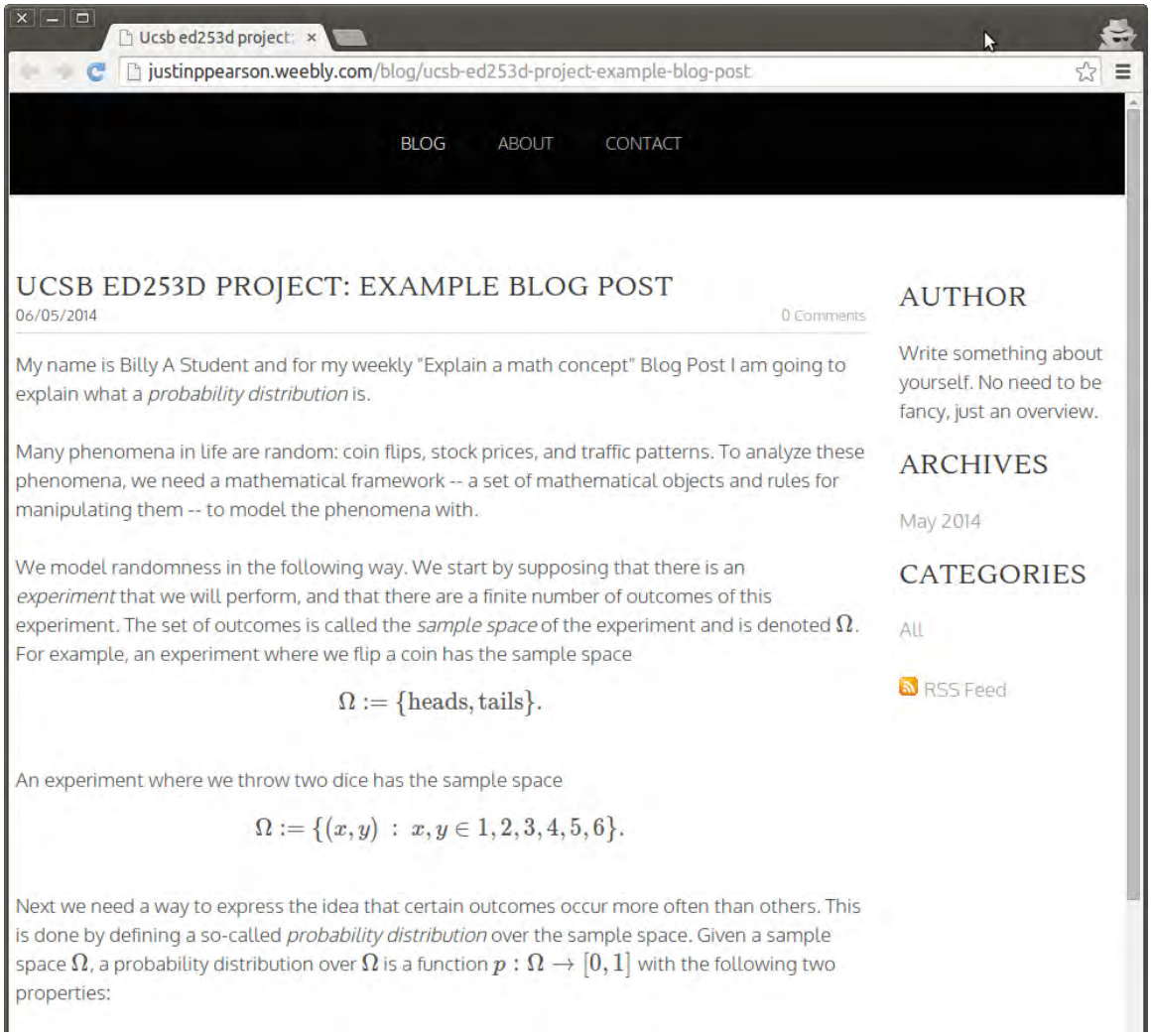
Using GauchoCast (a UCSB-tailored version of the Panopto screen-recording software), I recorded two lectures for the "flipped" component of the course. I enjoyed learning about GauchoCast and exploring what technologies are available for online learning.



The first video lecture from my course "Skills for Thinking Technically".

5.2.6 Example blog post

Practicing good mathematical and logical notation was an important component of my course. For this reason, my hypothetical students write weekly blog posts that explain mathematical or logical concepts, and they provide feedback to each other. To test the viability of this part of the course, I made an example blog post using the Weebly blogging platform. As with GauchoSpace and GauchoCast, I found it very useful to learn Weebly and expand my toolkit of pedagogical technologies.



An example of the weekly blog posts my hypothetical students would write.

5.3 Conclusion

ED 253D provided me with pedagogical theory, tried-and-true teaching tools from real UCSB professors, and the opportunity to practice these theories and tools by designing my own course. For example, Bloom's Taxonomy of Learning guided the structure to my course's Learning Pathway. My course's Learning Outcomes governed appropriate learning activities. And exploring pedagogical tools like GauchoSpace, GauchoCast, and Weebly vastly expanded the types of learning activities I could create. All these experiences support the educational theme that *an instructor must provide a purpose and a path to the student*.

CHAPTER

SIX

REQUIREMENT 4: TEACH A UNIVERSITY COURSE

A lecture from my class, Engineering 3: Introduction to Programming, Aug 2015

6.1 Executive Summary

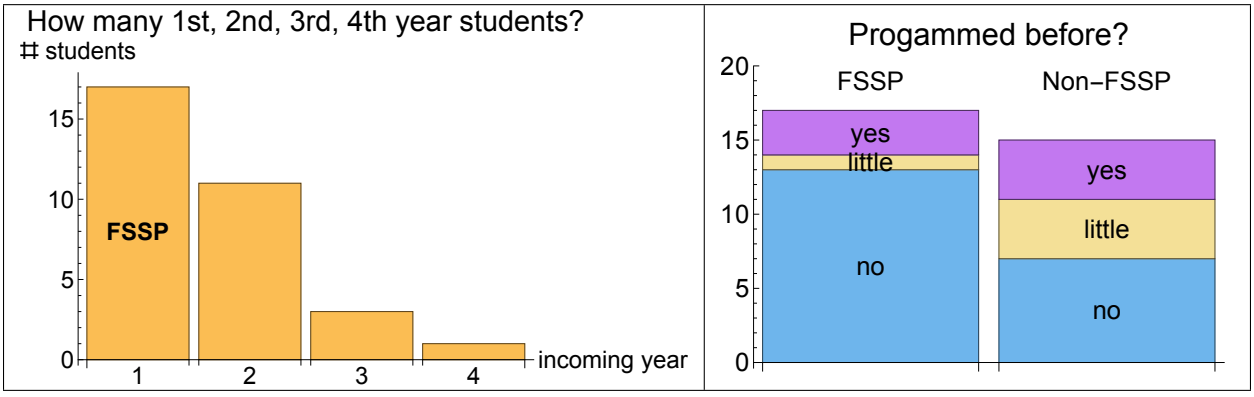
In Summer 2015 I co-instructed Engineering 3: *Introduction to Programming*, as a Teaching Associate. My co-instructor, UCSB Mechanical Engineering Professor Jeff Moehlis, had taught ENGR 3 several times and felt that the course could use a fresh set of eyes. The course material split naturally into two themes, and so Jeff and I split the course into two parts: I taught the programming aspects during the first three weeks of the six-week quarter, and Jeff taught mathematical applications for the last half. For my half, I designed the lectures, homeworks, quizzes, and the midterm exam. I participated in the second half of the class too, attending the lectures, holding office hours, and helping write and grade the final exam. Wanting to maximize the students' opportunity to practice programming, I instituted weekly "Hack Days", where lecture was conducted in a large computer lab. My co-instructor attended my lectures and provided guidance and feedback during our weekly meetings. We used feedback cards to assess the students' well-being and adjust the course as needed. The course went well: the students did well in the course and their feedback cards indicated that they felt they learned a lot about programming and enjoyed my enthusiasm.

6.2 Course Background

ENGR 3 is an introductory programming course for Freshmen engineers. Using the Matlab programming environment, students learn the basic datatypes and control structures that underlie any program. The course naturally divides into two parts: first, the students become proficient in basic programming constructs and patterns; second, the students learn mathematical concepts common in engineering — matrices, probability, and ordinary differential equations — and write programs involving these concepts. Consequently, it seemed natural to divide the course in half, with me teaching the Matlab-heavy first three weeks, and my co-instructor Professor Jeff Moehlis teaching the last three weeks.

6.2.1 Student backgrounds

Recall from *my experience TAing ENGR 3* that the main challenge of teaching ENGR 3 over the summer is the wide range of student abilities. Based on the lessons learned from TAing ENGR 3 the year before, we gave the students an informal entrance survey on the first day of class. This was proposed by Instructional Developer Lisa Berry: by collecting some data on students' programming backgrounds, we aimed to tailor our teaching efforts to suit the students' prior experience. Here are two relevant plots from that survey:



Entrance survey data.

Roughly half our 32 ENGR 3 students were in the Freshman Summer Start Program (FSSP), a bigger fraction than in 2014. Not surprisingly, most students had not programmed before. The 3 FSSP students with prior programming experience (purple box on left) had most likely taken programming classes in high-school.

The survey confirmed that we had a diverse student population and special care would need to be taken to keep students engaged while also not overwhelming them. Thematically speaking, the survey highlighted the importance of an empathetic instructor.

6.3 Course Design (Syllabus analysis)

Now we discuss the design of the course. Since this was my first time teaching a course, we kept it similar to what my co-instructor had done in previous years: lecture, TA discussion sections, weekly homework, office hours, weekly GauchoSpace quizzes, and GauchoSpace discussion forums. For specifics, please see the annotated syllabus in *Appendix 1*.

The following subsections examine noteworthy aspects of the course, and how they helped me grow as an instructor.

6.3.1 Cyber-Prof

My ENGR 3 lectures consisted mainly of demonstrating aspects of the Matlab programming language, punctuated by in-class exercises for the students to practice the language. However, it's boring when an instructor spends the whole lecture typing from behind the podium. I wanted a way to type on the computer *and* walk around the classroom, gesticulating and pointing. So I purchased a two-piece wireless keyboard and wireless trackpad. Using some spare screws, plastic, and an old wrist-guard, I was able to affix the keyboard, trackpad, and iClicker to my body:

The first day of class, after a motivational slideshow about the value of programming, I asked the students if they were ready to "Enter the Matrix... Laboratory" (a reference to the movie *The Matrix*; also, "Matlab" is short for "Matrix Laboratory"), and I put on the Cyber-Prof suit. The students were smiling and engaged.

Unfortunately, the Cyber-Prof suit didn't end up working out — my dress pants have weirdly-angled pockets so the keyboard kept falling out. Additionally, it was too hard to type with the keyboards shifting around. Consequently, I settled for typing on my laptop beside the podium, and jumping back and forth to the projector screen. Even though it didn't work out, I'm glad I tried the Cyber-Prof outfit: the students thought it was very cool while it lasted, and it got them excited.

6.3.2 Hack Day

In "The Road To Excellence" (2014), K. Ericsson reports three criteria for effective learning:

1. A well-defined task with an appropriate difficulty level for the particular individual
2. Informative feedback
3. Opportunities for repetition and corrections of errors

Traditional "broadcast-style" lectures don't accommodate these criteria. Although I peppered my lectures with in-class exercises for the students, I wished every student could be practicing Matlab in real-time during the lecture. That would provide them informative feedback (#2) in the form of correct answers or error messages, and permit them to correct their errors (#3), with an instructor present to help them.

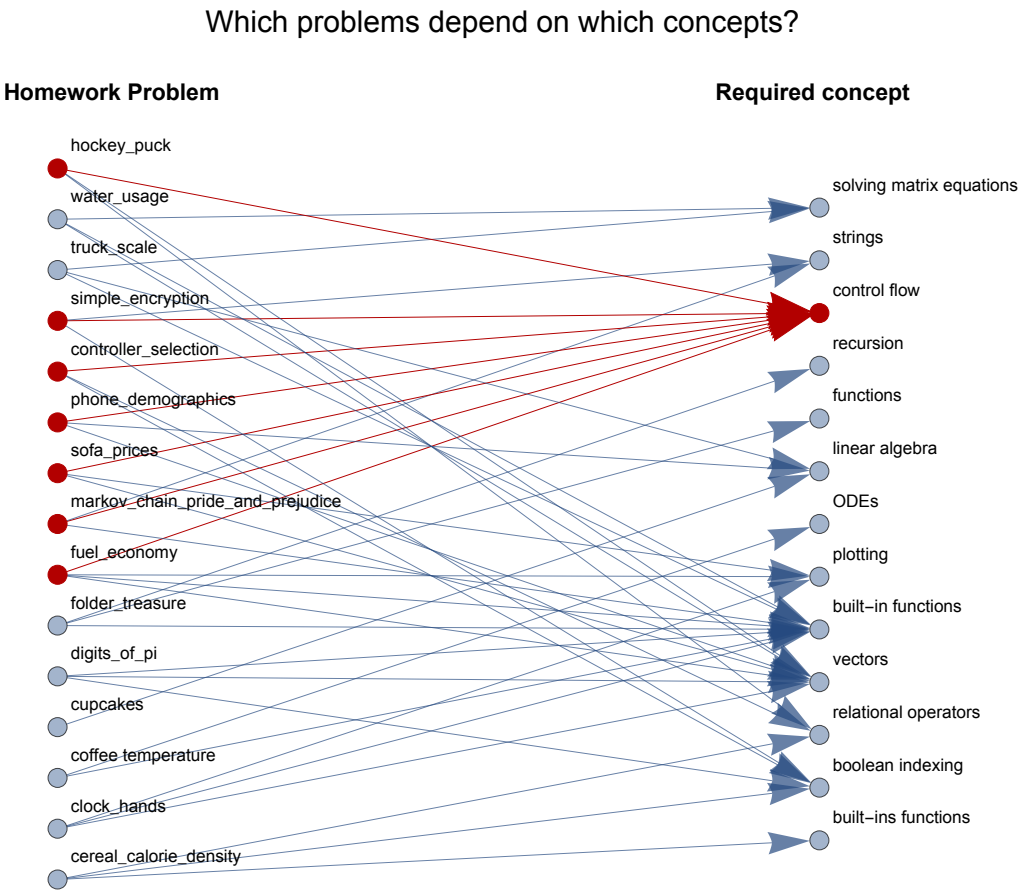
So, in keeping with the theme of "Students learn best when they struggle in a controlled environment," I dedicated the last lecture each week to "Hack Day": We met in a large 50-seat computer lab, and instead of a broadcast-style lecture, I would pose real-world engineering problems for the students to solve. The students would take 10 minutes to think about the problem, and my co-instructor Jeff and I would pace around the room helping students. Then we would reconvene and discuss students' solutions, and repeat the process.

I intended Hack Day to illustrate fun, hard, real-world engineering problems. I thought that would motivate the students to see what programming is good for. But most of the students wished that Hack Day would help them with their homework more directly. In hindsight, I should have known that would be the case, given the students' incentives: there was no grade associated with Hack Day. Perhaps I should have dedicated a portion of their grade to solving Hack Day problems in groups, as Professor Roger Freedman does with his undergraduate Physics courses. This speaks to the theme that "An instructor needs to empathize with the students' struggle" — their struggle to focus their energy on what matters (their grade).

Changing the grade weights to include Hack Day problems was out of the question — the syllabus is a contract between the instructor and the students. Instead, I re-designed Hack Days 2, 3, and 4 to have the student work in-class exercises that were more directly related to concepts necessary for their homework. The students indicated on subsequent feedback cards that they approved of the change, and felt the Hack Days were valuable.

6.3.3 Homework

In my preparation for the course, I wrote several dozen homework problems meant to illustrate fun, technical programming applications. Each problem exercised certain concepts, and in order to help me pick good homework problems based on what I had covered in lecture, I wrote a program to diagram which problems required which concepts:



A dependency graph of homework problems. For example, the red homework problems (left) all require the concept of "control flow" (right), so one needs to teach control flow before assigning those problems.

This was a lesson in the dangers of over-preparation. It became evident in the first week of the class that students new to programming felt lost and intimidated by the problems in Homework 1. The first homework was too ambitious — I was just so excited to show them all the cool things you can do with Matlab. But it required knowledge of several key elements of the language, and the students didn't have time to absorb all the necessary pieces. In hindsight, the students needed a slower, gentler introduction to programming. My TAs reported that the students in their sections felt unprepared and were unhappy. I felt badly for my students. I met with Jeff and the TAs to consider contingencies. We recovered somewhat by delaying the homework deadline by four days. This opened up two more sets of office hours for them. Also, we re-purposed the first Hack Day for hands-on practice. This actually worked really well, because students could get instant feedback from their own computers. In light of this lesson, I ended up re-writing new, easier homework problems for subsequent assignments and discarding most of the ones I had prepared. The students registered their satisfaction via feedback cards a week later.

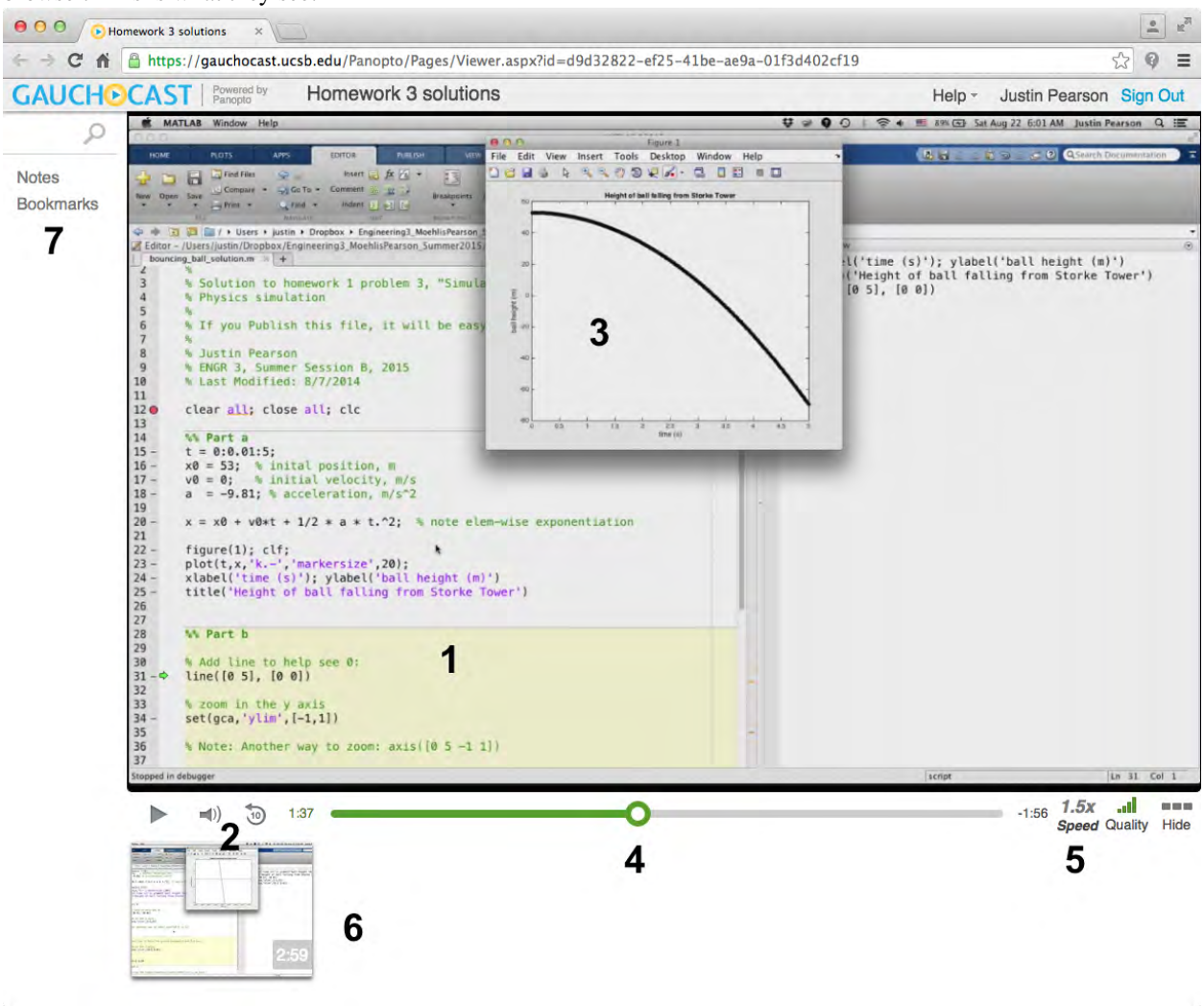
It's worth noting that some students (presumably ones with prior programming experience) indicated via feedback cards that they enjoyed the harder homework problems. Those students were in the minority, and not our target audience. Perhaps I should have offered some harder problems as extra credit, but this would need to be done tactfully so as not to favor the advanced students unfairly.

Ultimately, the main lesson I learned from "the Homework 1 debacle" ties back to the theme of empathy: an instructor needs to be able to empathize with the student. I had forgotten that many of my students had never programmed before, and many were used to the kinds of problems taught in high-school, where they are given a formula and must plug in some numbers. It is scary to start a college career, and many students were worrying whether they could succeed as engineers — or even succeed at college at all. The students needed a gentler transition, which we were able to provide in subsequent weeks of the course.

6.3.4 GauchoCast Homework solutions

In many courses, homework solutions are posted as PDFs to the course website (typically hosted on GauchoSpace). For ENGR 3, the homework solutions consist of computer code, which could be posted in the same way. However, posting solutions in code format is problematic for two reasons: First, computer code is hard to read — even if it contains code comments — so students are less likely to study it. Moreover, the code doesn't contain the *results* of running the code, so students can't easily see the correct answer unless they run the code themselves. Second, PDFs and computer code files are easily uploaded to homework-solution-sharing websites like coursehero.com. These websites pay students to upload their courses' homework and exam solutions, and other students pay coursehero.com for access to these materials. To prevent future ENGR 3 students from buying homework solutions, we wanted to disseminate homework solutions in a format that would be difficult for students to upload to coursehero.com.

We provided homework solutions in a new way: I recorded screencasts of the homework solutions using the Panopto screen recorder, and hosted them using UCSB's GauchoCast video-hosting service. Each 3-10 minute video solved one homework problem. In each video, I show the code, explain it, and run it. Students view the videos in their web browser. This is what they see:



The Panopto video player plug-in has many nice features (as indicated on the figure):

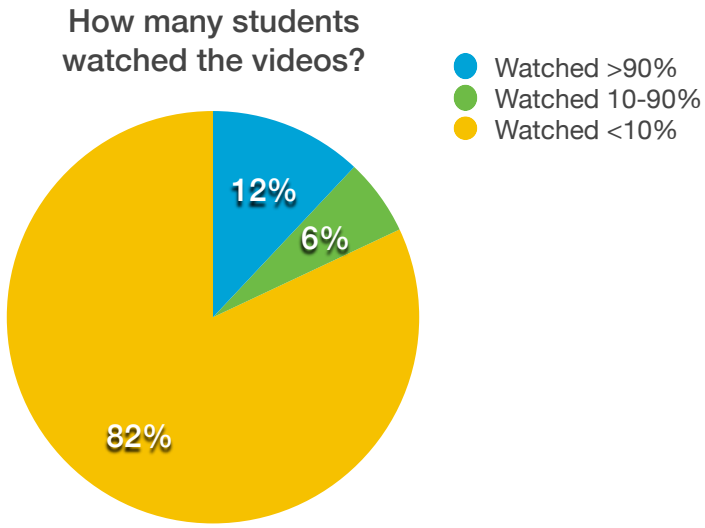
1. students can see the code on my screen
2. students hear me explain the code, which is easier than scrutinizing it themselves

3. they see the results of me running the code step-by-step, e.g., this graph
4. they can skip around in the video, rewind it, etc.
5. they can play the video faster or slower than real-time, to match their individual learning speed
6. thumbnail video frames help outline the video
7. students can bookmark parts of the video and can leave comments at certain times

Moreover, the video files are streamed from UCSB's servers and are difficult to download and too large to distribute on coursehero.com or other file-sharing sites.

We hoped that these videos would make the homework solutions more accessible for students, as well as limit their dissemination to future students of ENGR 3.

Based on GauchoCast metadata, 12% of the class watched the videos, 6% watched a portion of the videos, and 82% didn't watch the videos:



See *Appendix 8* for a more in-depth analysis.

In light of this data, I would not post homework solution videos in the future. They are fairly easy to record, but it seems that the vast majority of students don't utilize them. The benefit of being hard to copy could be achieved in other ways, e.g., password-protected PDFs. Furthermore, I note that the two lowest-performing students were among the most avid viewers. This may have been last-minute cramming, or might be a more dangerous indication that the videos lull viewers into a false sense of knowledge. In hindsight, I should have asked the students what they thought. This is an opportunity for me to grow as an educator: Instead of poring over the viewership data like a good engineer, it would be more characteristic of an *empathetic instructor* to poll my students directly about the videos' usefulness.

6.3.5 Feedback cards

Feedback cards were immensely useful in tuning the course to suit the students' needs. During my 3 weeks of lecturing, I gathered in-class feedback twice:

- Aug 12: at the beginning of the second week, after the first homework was due.
- Aug 19: at the end of the third week, the day before the midterm exam.

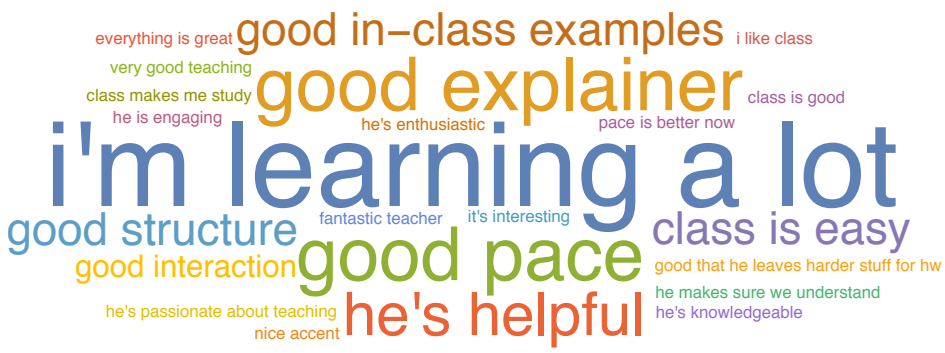
Here are the questions I asked:

1. Are you in the Freshman Summer Start Program?

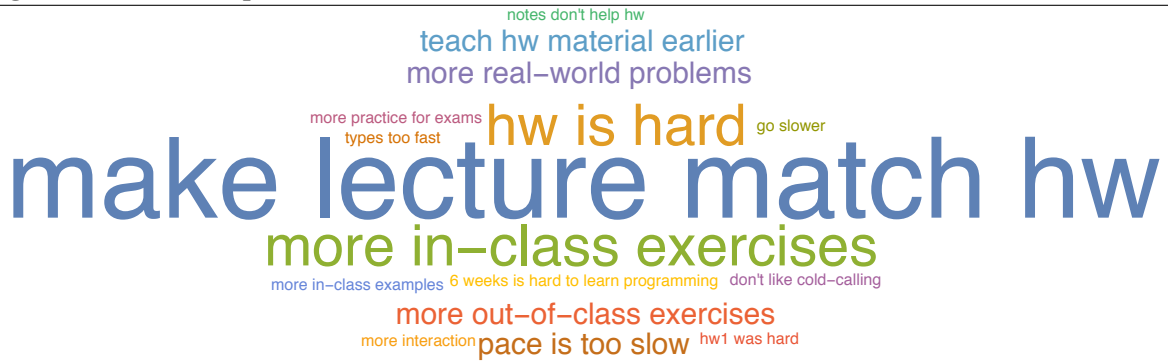
2. Going into what year?
3. Have you programmed before?
4. Speed of lecture: want faster, slower, or OK?
5. In-class exercises: want more, fewer, or OK?
6. What's going right with the class?
7. What would you improve with the class?

In the first set of feedback cards, the students felt that the class was going at a good pace, but the vast majority of students felt that Homework 1 was too hard and the lecture didn't prepare them for it. Their open-ended feedback is represented in this word cloud, where the size of the phrase indicates how many students voiced it:

Aug 12: What's going well?



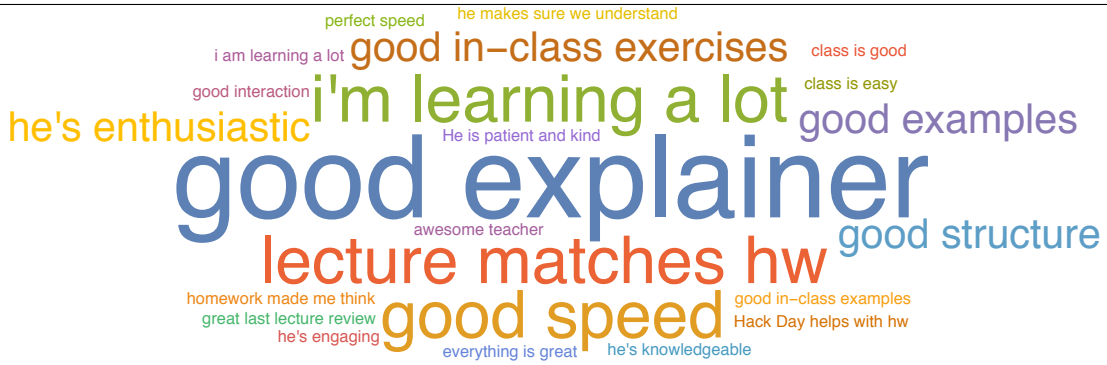
Aug 12: What needs improvement?



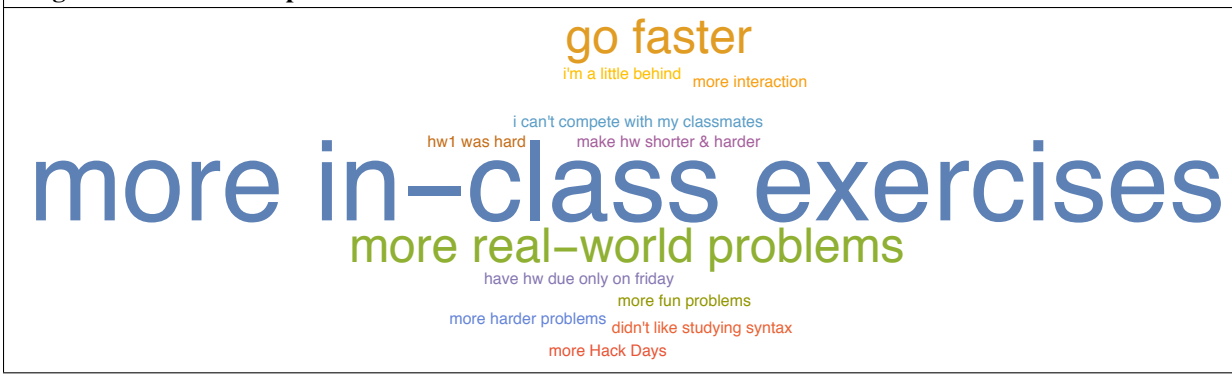
We responded to this by making subsequent homeworks easier and doing more in-class exercises of the material in class. Each exercise consisted of the students working independently for 3-5 minutes on a short programming problem, then discussing it. Moreover, I reworked the Hack Days to incorporate in-class exercises that were vaguely related to the homework, instead of using them to explore harder extensions to the material.

The students appreciated these changes. The second set of feedback cards indicated that they were satisfied with how the lecture matched the homework, but they wanted even more in-class examples and exercises:

Aug 19: What's going well?

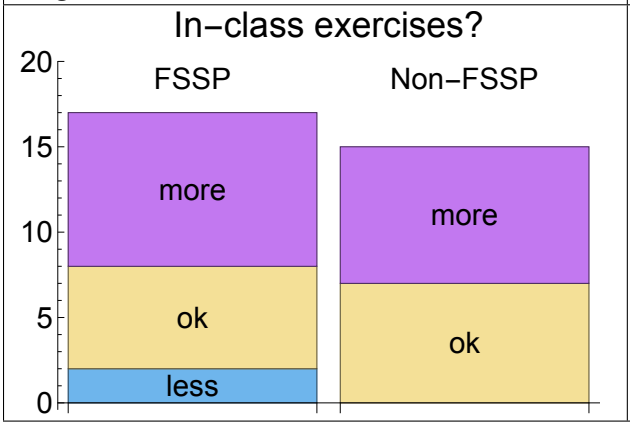


Aug 19: What needs improvement?

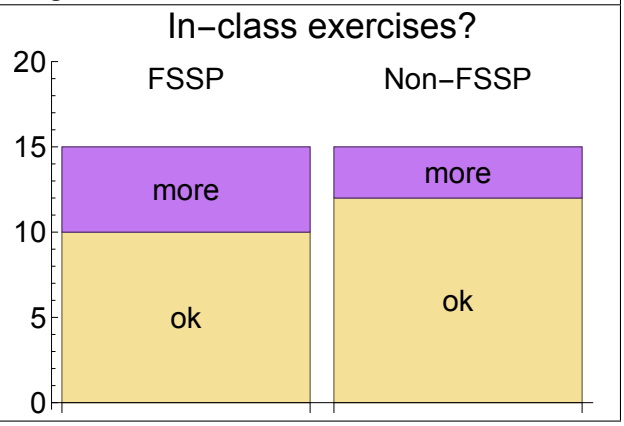


Here is a more quantitative comparison of their desire for more in-class exercises:

Aug 12: Want more/fewer in-class exercises?

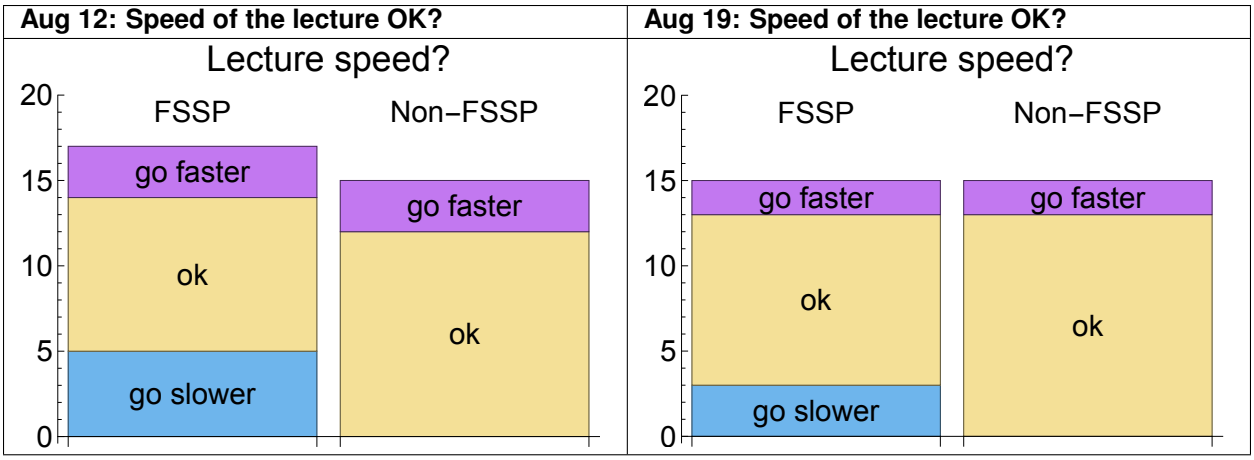


Aug 19: Want more/fewer in-class exercises?



I was pleased that we were able to satisfy several of the students, and pleasantly surprised that the students wanted even more exercises. I had expected that, since the in-class exercises require effort on the students' part, they would prefer to just have me lecture. The optimistic interpretation is that the students wanted the opportunity to practice the material in order to learn it better. The pessimistic interpretation is that the students wanted to "run the clock" with easy in-class exercises instead of learning new material for which they would be responsible. I personally believe that most students aligned with the optimistic view. My sense is that students like it when they can practice problems that are similar to the exams, and this was their main impetus for wanting more examples and exercises.

The feedback cards reassured me that I was going at an appropriate speed: In both sets of feedback cards, the students were reasonably happy with the pace. About the same number of students wanted me to go faster as the number that wanted me to go slower:



Feedback cards have the additional benefit of giving students a voice and helping them feel validated. I made a point to tell the students the main themes present in their feedback, to reassure them that I read them and that I cared. I empathized with their struggle.

6.4 ESCI Teaching Evaluations

My teaching evaluations are summarized in *Appendix 5*. I observe:

What worked:

- Students felt I was prepared, enthusiastic, helpful, and clear.

What needs improvement:

- *Adapting to students' abilities.* A third of my students felt that I was only "good" at adapting my teaching to varying student abilities. (65% rated me "superior".) This may be due to the variation in student abilities: It is difficult to teach at every student's optimal level. Still, I could improve this by possibly asking for more feedback and posing more in-class questions to assess whether the students are following along.
- *Fairness of assignments.* 26% of my students felt our grading procedures were only "good", while 10% felt they were "average". (61% rated the fairness "superior", and one student rated it "poor".) I suspect these relatively low scores derive from the automatically-graded GauchoSpace quizzes. The GauchoSpace quizzes expect answers to have a very specific format, so it is difficult to pose programming questions. For example, Matlab doesn't care about extra spaces in your program, but GauchoSpace quizzes do. We tried to combat this by placing a formatting guide at the top of each quiz that would help them conform to GauchoSpace's expectations. But we still had to re-grade several quiz questions that had wrongly penalized students. In the future, I would consider using GauchoSpace quizzes only when a question has a specific numeric answer.
- *Class notes not helpful.* 35% of my students felt that the course notes were only "good", and 6% felt they were "average". (The remaining 59% thought they were "superior".) I thought the students would like how I formatted the course notes: nicely-formatted Matlab code with explanations. I interspersed little exercises in my notes and left the answers blank for the students to fill in. It's possible that students found this bothersome. In the future, I would like to figure out a way to assess whether the notes are useful to the students, and how they are using them. I expect that most students used them only as reference material, not as a lesson to study. Perhaps they needed guidance in how they should interact with the lecture notes.

I was pleased with my students' open-ended reviews:

- Great passion & dedication for work.
- This course, albeit fast, was able to pack a lot of material into not that much time. Justin was extremely helpful in class and outside of class, and his lectures improved after he asked for class feedback. Overall, he is very

approachable and an amazing & clear instructor.

- First half of class was really hard to grasp at first.
- Absolutely amazing professor. Should 100% be a full time professor here. Extremely passionate and open to help students.
- Good speaking voice, clear and precise. Notes in class also very thorough, but the ones posted online are not as good.
- I found Professor Justin Pearson to be a fantastic instructor while taking this course. He was able to clearly and effectively teach the course at a challenging yet manageable pace.
- Justin and Jeff are awesome teachers who really get me enthusiastic about programming. The first hw was very hard but also very rewarding!

6.5 Conclusion

Teaching a class took an incredible amount of work and was immensely gratifying. Guiding students through a potentially frustrating subject was electrifying, and I loved sharing my enthusiasm with them. Along the way, I learned several important lessons:

- Assigning an **introductory survey** helped me learn my students' backgrounds and tune the course appropriately.
- After an initial rough start with Homework 1, I learned to **assign homework of appropriate difficulty**.
- I collected **student feedback** often and adjusted my course in response, for example by incorporating more in-class exercises.
- **GauchoCast video homework solutions** weren't as helpful to the students as I had hoped, and I should have polled the students to determine that earlier.
- To maximize student interest, **learning activities should prepare students** for doing well on homeworks and exams (or whatever they are graded on). For example, I learned to use Hack Days to emphasize core concepts instead of exploring harder applications.

I feel very lucky to have had the opportunity to teach a university-level course, and I truly look forward to teaching subsequent courses at some point.

CHAPTER

SEVEN

CONCLUSION

I have benefited enormously from the CCUT program, and the final step — preparing the CCUT portfolio — allowed me to reflect on what I've learned. Here are some of the ways that the CCUT helped me grow as an instructor, expressed in the framework of the three themes of my teaching philosophy:

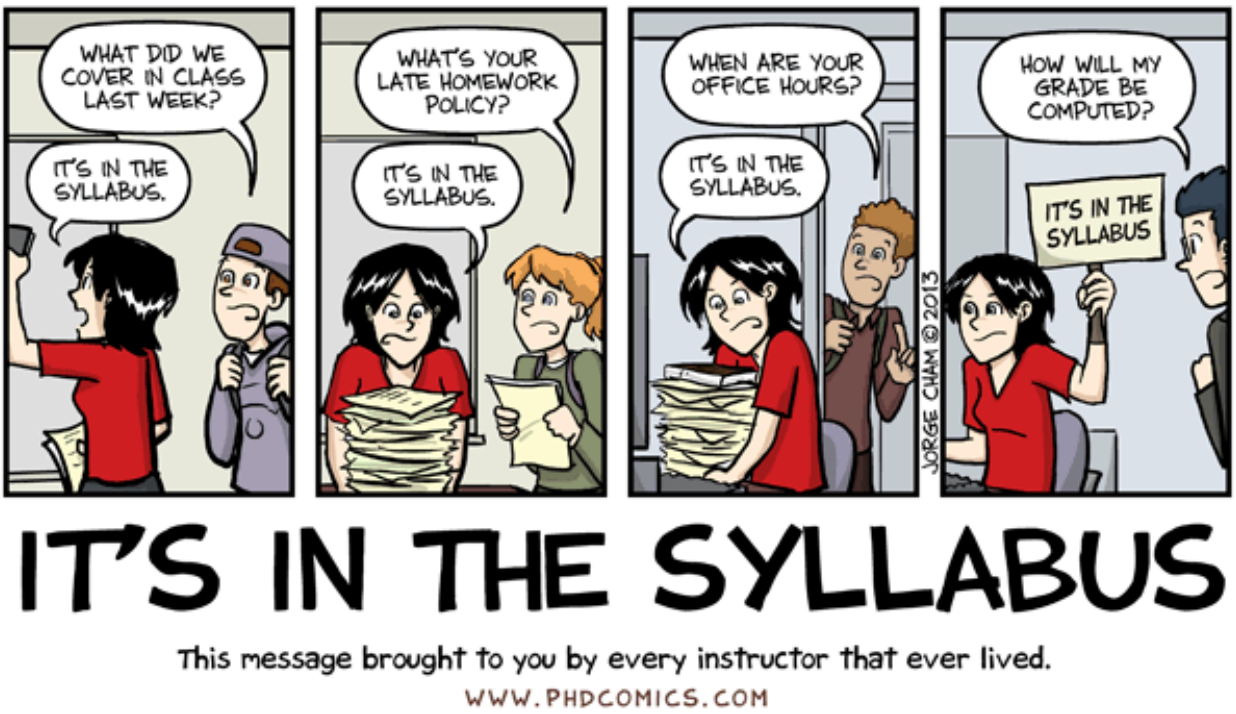
- **Purpose and path.** The CCUT helped me learn how to motivate my students and provide them with a structure to learn in. In my education class, I practiced writing Learning Outcomes and studied Bloom's Taxonomy; this greatly facilitated my design of ENGR 3. Familiarizing myself with other software tools like GauchoSpace and Panopto provided myriad ways of adapting to students' needs. And when teaching ENGR 3, I had some opportunities to redirect the flow of the course to better align its path with what motivated the students (their grades). These tools and experiences will help me motivate and guide many students to come.
- **Struggle.** Throughout the CCUT I had multiple opportunities to practice letting my students struggle an appropriate amount. In my ENGR 3 class, I used feedback cards to design homeworks and in-class exercises of suitable difficulty. In the SST, I found that students could effectively modulate each other's struggles by working together. And as I saw when TAing the pendulum lab, the appropriate level of struggle varies from student to student, and an instructor needs empathy to strike the right balance. I feel confident in my ability to guide students toward healthy, productive learning and away from needless frustration.
- **Empathy.** Working with Seniors in ECE 147C, Freshmen in ENGR 3, and high-school students in SST helped me appreciate the wide diversity of students I will encounter as an instructor. In listening to their struggles and helping them become better engineers, I became better at empathizing with many types of students. While pursuing the CCUT, I helped frustrated high-schoolers program their robots, bewildered Freshmen debug their Matlab code, and jaded Seniors right their fallen pendulums. Each case required a different form of supporting and empathizing with the student, and the variety of these interactions has improved me as an instructor.

In conclusion, the CCUT was a highlight of my graduate career, and I am so pleased to have had the opportunity to pursue it. It focused my pedagogical philosophy and imbued me with educational skills and structure that will guide me in a lifetime of teaching.

CHAPTER

EIGHT

APPENDIX 1: ANNOTATED SYLLABUS



PhD Comics, #1583

ENGR 3 — Introduction to Programming

Summer Session B 2015: Mon Aug 3 – Sat Sept 12

MTWR 9:30–10:50 am in Phelps 3505 (MTW) and SSMS 1301 (R)

Course Description

General philosophy of programming and computational problem-solving. Students will be introduced to the programming language Matlab. Specific areas of study will include algorithms, basic decision structures, arrays, matrices, and graphing.

Copied from the UCSB Course Catalog.
I would've preferred a more engaging opening

Instructors

This course has two instructors: Graduate student Justin Pearson from Electrical and Computer Engineering will teach the first three weeks of the course, focusing on the Matlab software environment. Professor Jeff Moehlis from Mechanical Engineering will teach the last three weeks, emphasizing mathematical applications.

Jeff Moehlis	Justin Pearson
moehlis@engineering.ucsb.edu	jppearson@ece.ucsb.edu
Office Hours: Tues 11–12 Eng II 2341	Office Hours: Mon 11–12 HFH 5152

Teaching Assistants

David Adams	Mishel George
adams@mat.ucsb.edu	mishel@engineering.ucsb.edu
Office Hours: Thurs 11–1pm Phelps 1526	Office Hours: Wed 1:30–3:30pm Phelps 1526
Disc sxn: Tues 2:00pm–3:20pm Phelps 1529	Disc sxn: Wed 3:30pm–4:50pm Phelps 1529

Grading

Considered having computer-based exams. Would align w/ the L.O. "you will learn Matlab programming," but logistics were too difficult.

Final Exam: 40% (Thursday, September 10)

Midterm Exam: 30% (Thursday, August 20)

Homework: 20%

Quizzes: 10%

HW and Quizzes are weighted less bc we assume the students are working in groups and presumably sharing their answers.

Hack days

Significant departure from how ENGR 3 is usually taught.

Every Thursday (except for the Midterm and Final above) is **Hack Day**: lecture is held in **SSMS 1301**: a large computer lab in the Social Sciences & Media Studies building. Each Hack Day presents a large, real-world programming problem that we work through as a class. It's an opportunity for you to practice programming with an instructor present to help you.

Students' feedback cards: Hack Day was a positive experience. They enjoyed being able to run code in real-time alongside the lecture.

We considered using iClickers, but after some internal tests, decided not to. Proficiency in Matlab is a key learning objective, and is better achieved via written code exercises than via mult. choice questions.

Course materials

Matlab. If you have a computer, you should buy Matlab. You will find this much more convenient than the campus computer labs, especially since they close at night (when the homework's due). The Student Version is ∼$100 from `http://www.mathworks.com/academia/student_version/`

Campus computer labs with Matlab:

- College of Engineering labs:

  `http://www.engineering.ucsb.edu/eci/services/labs/`

  CoE labs require a login, see `https://accounts.engr.ucsb.edu/create`

- Collaborate Labs:

  `http://collaborate.ucsb.edu/home`

GauchoSpace. We use GauchoSpace (`https://gauchospace.ucsb.edu`) to manage this course. Log in with your UCSB NetID. If you are registered for the course, it will appear on your GauchoSpace home page. Course materials will be posted to GauchoSpace. It is your responsibility to check GauchoSpace for new materials and assignments.

Textbook. There is no required textbook, but with a full college career ahead of you, I'd recommend a good Matlab reference: *Getting Started with MATLAB: A Quick Introduction for Scientists and Engineers* (2009) by Rudra Pratap, $15 rent / $38 buy on Amazon.

Textbook optional. We already made them buy Matlab. We posted our own lecture notes and other written modules to GauchoSpace.

Homework

Weekly programming assignments that require Matlab will be posted and collected via GauchoSpace. You may discuss problems with classmates, but the work that you submit must represent your own effort and understanding. Do not copy from others. The due date is typically 11:55PM on the specified date. No late homework will be accepted.

The TAs reported minor plagiarism in hw2. I sent a stern email mid-quarter reminding them to not copy from each other.

Quizzes

Short weekly quizzes will be administered through GauchoSpace. These are mainly to keep you sharp and current on the material. You may use Matlab while taking a quiz, but you must take quizzes alone — no discussing with classmates. Once a quiz is posted on GauchoSpace you may attempt it at any time before the due date. You will have 2 chances to take each quiz (but the questions are selected randomly by GauchoSpace so each chance will be different); your highest grade will be recorded by GauchoSpace. No late quizzes will be accepted.

Quizzes were meant to be easy — keep them practicing the syntax.

Exams

One midterm exam and one final exam will cover material from lectures and homework assignments. Exams are in class, on paper (not on the computer). No cheat-sheets, phones, calculators, or friends

— exams must be taken individually. The final exam is cumulative, with emphasis on the second half of the class. There will be no make-up exams given.

Hints

Quizzes and homework are designed to help you learn the material. If you learn the material, you should do well on the exams. With 70% of your grade depending on the exams, there is no way to do well in this course without doing well on exams. The best way to learn the material is by doing the quizzes and the homework on your own.

Each week in this course builds on the material of the previous weeks, for both the programming and the math skills. Review the lecture material, do the quizzes and homework, and ask questions whenever there is something you do not understand. We offer office hours to help you to learn the material. Take advantage of these opportunities. **Do not wait until the night before the exams to try to learn the material. It will not work.**

While class attendance is not mandatory, experience shows that students who do not attend class do not receive satisfactory grades. Therefore, your attendance in lecture will be crucial to your success in the course.

Academic Misconduct

We will be vigilant in prosecuting cheaters. If you are caught cheating, you will receive an F for the course, and you may be referred to academic judiciary at the Office of Student Life. The standard penalty the Office of Student Life issues for plagiarism is a 2-quarter suspension from UCSB, although it could be more severe.

Misc

If you are a student with a disability and would like to discuss special academic accommodations, please contact me at your earliest convenience.

CHAPTER

NINE

APPENDIX 2: TA VIDEO CONSULTATION

TA DEVELOPMENT PROGRAM
INSTRUCTIONAL DEVELOPMENT, UC SANTA BARBARA
1130 Kerr Hall 805.893.4289 http://oic.id.ucsb.edu/ta-development-program

October 13th 2014.

TO: Justin Pearson, Department of Electrical & Chemical Engineering, UC Santa Barbara
FR: Kim DeBacco, Instructional Consultant, UC Santa Barbara
RE: Classroom Taping: *ECE 147c;* recorded on May 1st 2014 (spring quarter).

CONTEXT:

Course:	*ECE147C/ME106A* — Control Systems Design Project/ Advanced Mechanical Engineering Laboratory
Enrollment:	12 ECE, 1 ME, 3 UCSB Extension (Foreign Exchange students)
Location:	SH 1430; Lab: HFH 3120
Class Time/s:	Lecture: Tues/Thur.: 12:30pm - 1:45pm
Topic:	Designing a Controller for a Physical System

This taping records you giving a guest lecture on controller design for this class.

INITIAL DISCUSSION:
Prior to viewing the recording of your class we discussed contextual details and background information. You mentioned that you had had little teaching experience as an instructor of record, but you cited a number of influences, including an exceptional high school teacher, being a TA for ECE 147c, tutoring elementary school students and giving conference presentations.

We also considered your strengths and goals for the class. In terms of your teaching strengths, you cited:

- your consistency in any teaching session for outlining the goals of the class at the beginning and revisiting them at the end of class;
- your ability to explain the conceptual material in clear, succinct, accessible way;
- your efforts in connecting new material in the course with what has been previously covered.

Your goals for this class were notably content-focused. Your intention was to help students learn "how to convert so-called closed loop specifications into specifications on the open loop transfer function."

Having already viewed this recording prior to our discussion, you had also identified aspects of your teaching that you would like to improve.

- You noted the students didn't ask any questions.
- You would like "a way to assess their learning mid-lecture".

VIEWING DISCUSSION

- *Self presentation; presence:* As we watched your taping, it was clear you have the appearance and confident manner of an experienced instructor. However, we both noted your evident tendency to talk too fast, and it is worth reflecting on why this is so. Do you feel you have lots of content to cover or present? Your rapid fire speech occasionally left you breathless and you can be heard drawing breath or sighing deeply several times during the recording. (See for example the 17 minute mark).
- *Presentation skills:* You were clearly very well prepared, with detailed teaching notes in hand. Your presentation centered on writing and drawing diagrams on the blackboard

accompanied by your (notably quick and succinct) explanations. Do your students need more time to digest and make sense of this new material?

- *Board work:* As the class progressed, you moved across the board from left to right, organizing the material in segmented columns. I wondered if you needed to increase the size/visibility of your handwriting? We also discussed your positioning at the board. Your back was to the students for most of the session. I suggested you try pivoting to the left on your right foot, so that you face your students while still being able to write with your right hand. This will enable you to have better eye contact with the class and individual students.
- *Student engagement:* We noted that even though you regularly asked the group: ("Any questions? Okay."), you gave them very little time to compose a question to ask. Your presentation was so very assured and confident, but that may have also contributed to their reluctance to ask questions. My recommendations included:
    - Consider focusing more on your students and less on the content.
    - You need to plan not just the content but also *the process* of the class. Prepare precise questions you will ask your students; carefully plan and stage pair/group work activities for predetermined periods of time with a whole group check-in to follow. When lecturing to small groups, can you find a point in each class to integrate a "think-pair-share" activity (which you mentioned on your preparation sheet)?
    - Stop and use more silence when you are at the board, asking students to suggest the next step or symbols in the sequence. Take a breath.
    - Prepare the board work (notes and diagrams) before class on paper sheets for display under a document camera. With a document camera, you can face the class directly and you can cover and reveal lines of content and drawings as you go. This will save you the energy of writing and give you more time and better positioning for watching and listening more actively to your students, so as to check their understanding.
    - Consider using some of the "classroom assessment techniques" to check students' learning and understanding (See the accompanying handout).

SUMMARY

The standout characteristics of this guest lecture were your review of the previous class material, your enthusiasm for the content, your accurate detailed explanations, and your ability to connect the abstract content using real world examples. Your students were attentive, if notably silent and passive; your presence was "teacherly" and confident.

YOUR TEACHING AND IMPROVEMENT GOALS:

Based on our viewing and discussion, you said that in the future you would focus in particular on strategies for increasing student involvement during the lecture, and more planning and monitoring of "spacing and pacing" during the lecture.

I am very happy to offer a classroom observation of your teaching in future, whenever you are ready to reflect again on your teaching. Justin, thank you for this opportunity to view one of your first lectures and discuss your teaching!

If you would like to arrange for additional video tapings to improve your instructional skills, conference presentations or job talks, please email us at tavideo@id.ucsb.edu. Or, you may arrange for a consultation without a tape to discuss teaching and learning issues.

CHAPTER

TEN

APPENDIX 3: FACULTY MENTOR LETTER

Me and my co-instructor Professor Jeff Moehlis, right after submitting the ENGR 3 final grades! Sep 2015.

Department of Mechanical Engineering
Engineering II Building, Room 2341
University of California, Santa Barbara
Santa Barbara, CA 93106

September 12, 2015

Certificate in College and University Teaching
Faculty Advisory Board

Dear CCUT Advisory Board,

I'm writing this letter to confirm that I mentored Justin Pearson during the Spring and Summer of 2015 while we designed the course Engineering 3: Introduction to Programming for Summer Session B.

Early on, we decided that it would be natural for Justin to teach the first three weeks of the class, and I would teach the last three weeks. Justin's half introduced the Matlab programming language, and my half revolved around applying Matlab to solve problems involving higher-level mathematics, e.g., linear systems, differential equations, and probability.

Justin and I met weekly and discussed every aspect of the course: he proposed a course schedule, adapted our syllabus from the previous year, designed homework assignments, wrote his own lectures, and wrote the midterm exam. He proposed using the iClicker student-response system, and prepared a short demo for me. (Ultimately we decided to forgo the iClicker in favor of traditional methods of calling on students.) He designed and conducted "Hack Days" — one day a week, lecture was held in a large computer lab, where the lecture would revolve around a programming problem that the students would tackle themselves, with instructor guidance and help.

While the course was underway, our meetings revolved around the status of the course — mainly the pace of the lectures, and which material we had to cover to prepare the students for the homework. The first homework assignment was a bit too ambitious, and Justin reduced the difficulty of subsequent homeworks and slowed the pace of his lectures, opting for more in-class examples. Students indicated via feedback cards that they felt much more comfortable with this change.

At the end of the class, we discussed grade distributions and set the curve for the class together.

Justin connected with the students. Although he decided to forgo using the iClicker, he was intent on having an interactive classroom and assessing the students' understanding of the material during lecture. He wrote down the students' names on cards to randomly call on them during class, and he polled them twice using short free-response feedback cards. Based on the first set of feedback, he re-designed his Hack Days to focus on fundamental concepts rather than advanced programming problems. The second set of feedback cards indicated the students appreciated this modification.

It has been a pleasure mentoring Justin. He clearly has an enthusiasm for teaching and a passion for reaching the students.

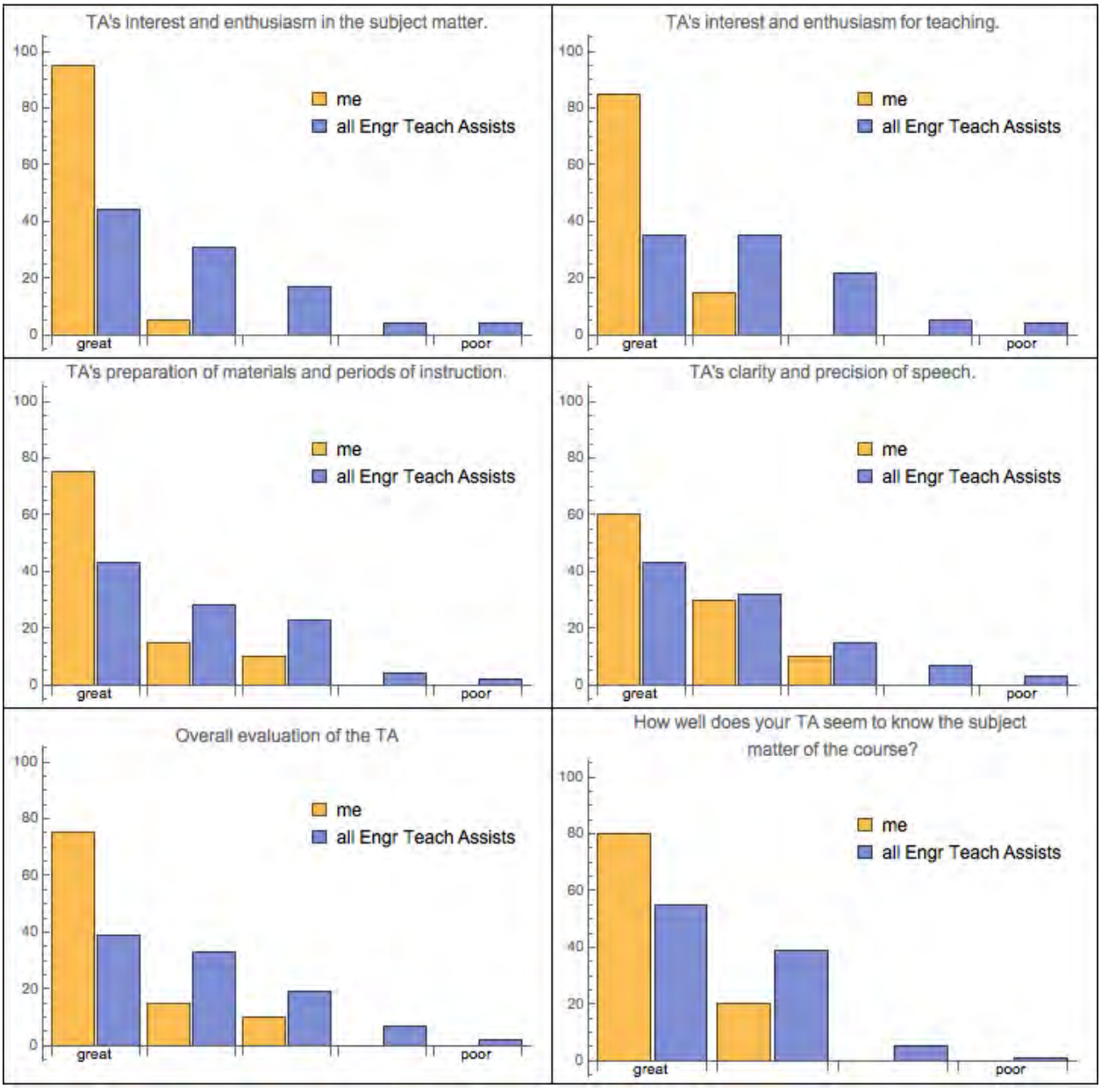
Sincerely,

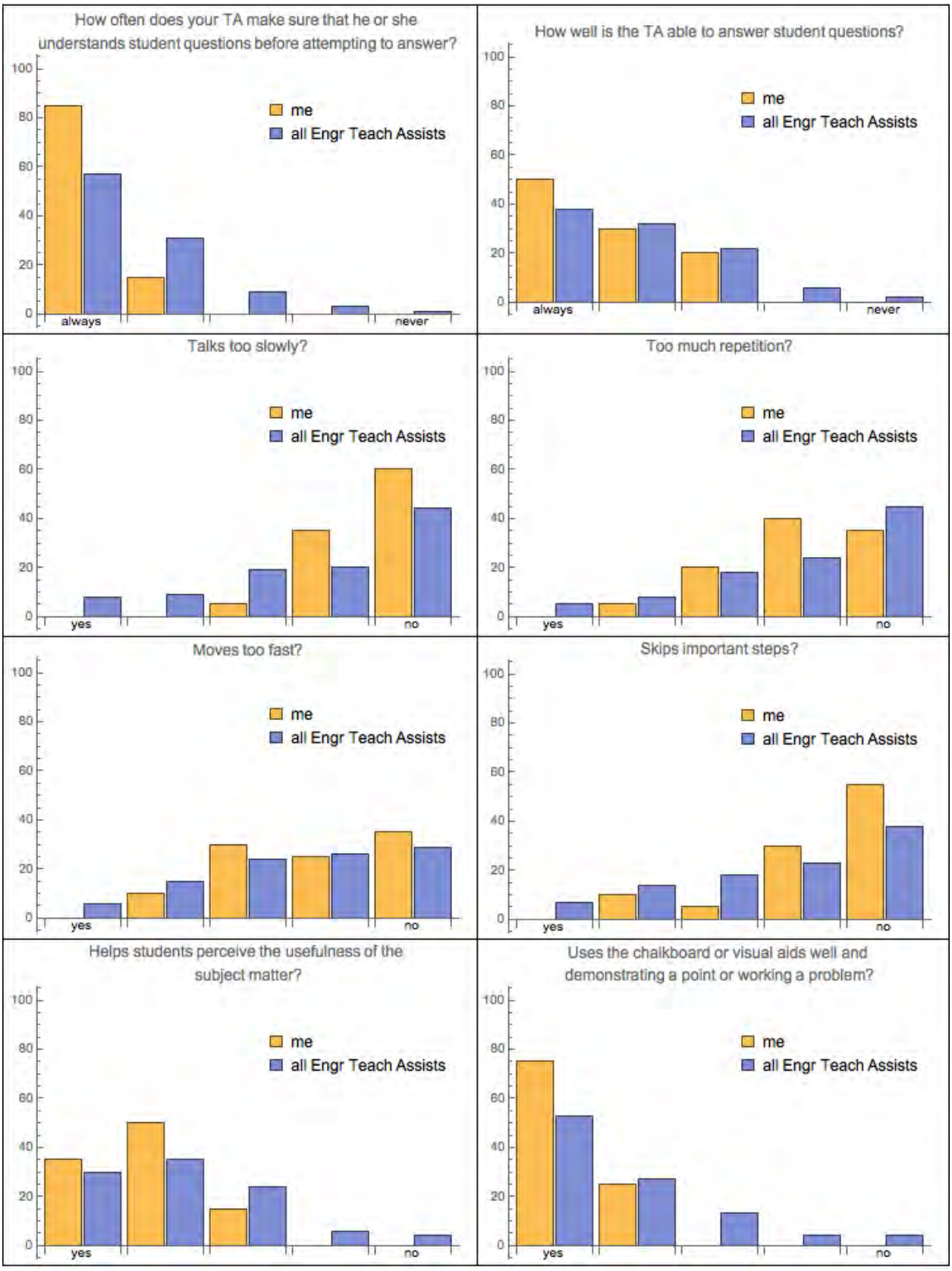
Jeff Moehlis

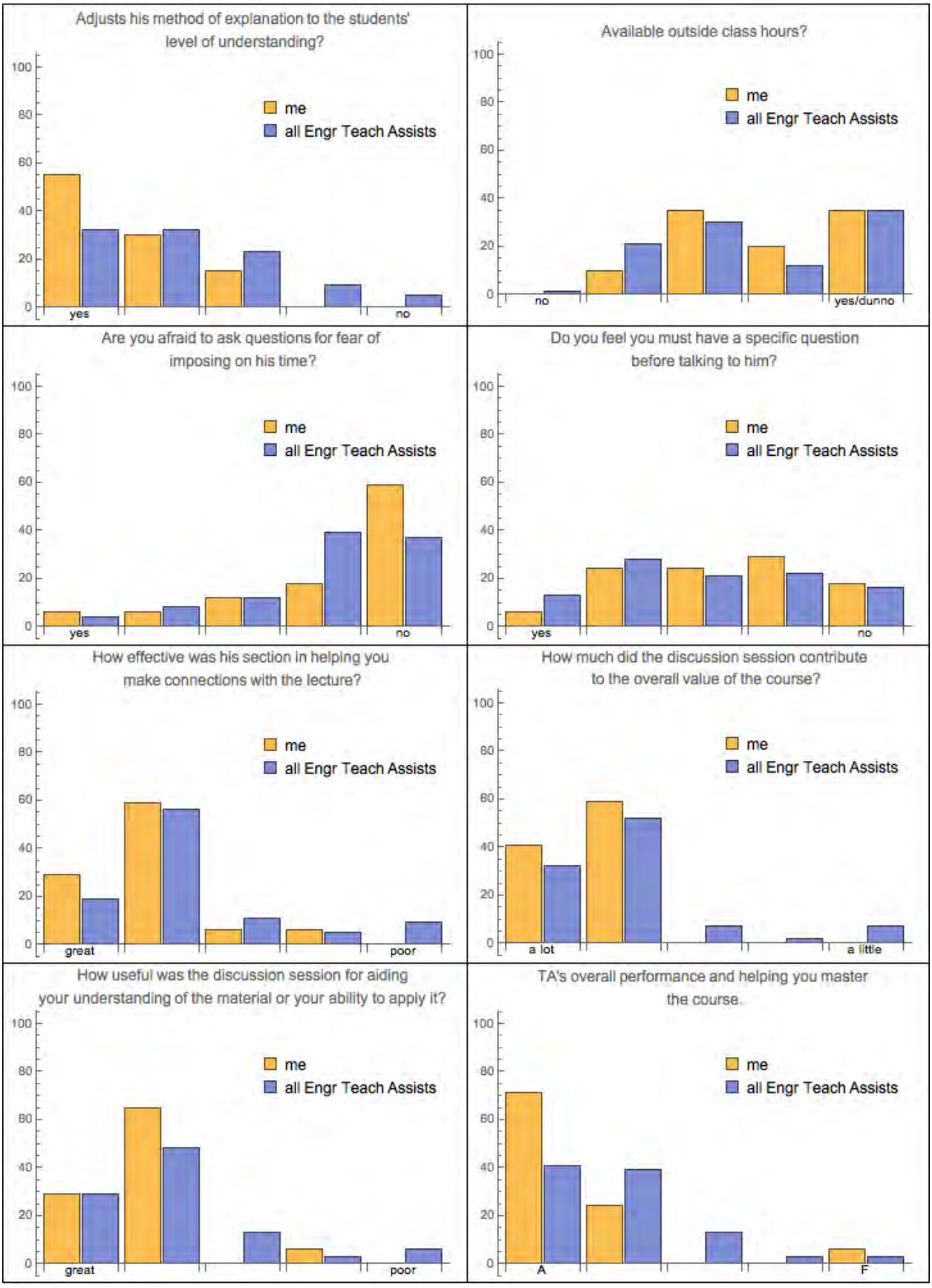
CHAPTER

ELEVEN

APPENDIX 4: ESCI TEACHING EVALUATION DATA (TA, ENGR 3)



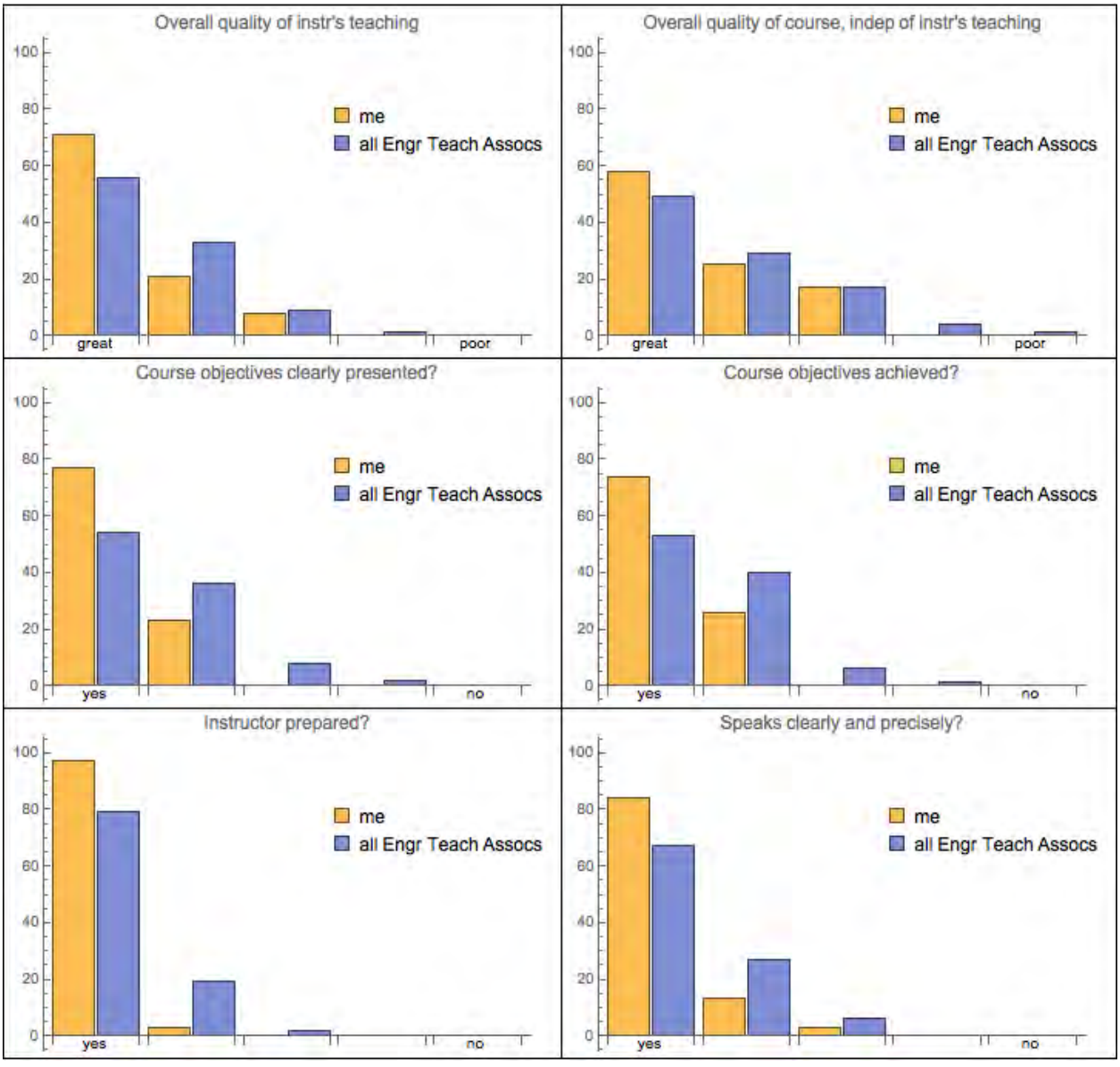

How often does your TA make sure that he or she understands student questions before attempting to answer?
me
all Engr Teach Assists
always
never
How well is the TA able to answer student questions?
me
all Engr Teach Assists
always
never
Talks too slowly?
me
all Engr Teach Assists
yes
no
Too much repetition?
me
all Engr Teach Assists
yes
no
Moves too fast?
me
all Engr Teach Assists
yes
no
Skips important steps?
me
all Engr Teach Assists
yes
no
Helps students perceive the usefulness of the subject matter?
me
all Engr Teach Assists
yes
no
Uses the chalkboard or visual aids well and demonstrating a point or working a problem?
me
all Engr Teach Assists
yes
no

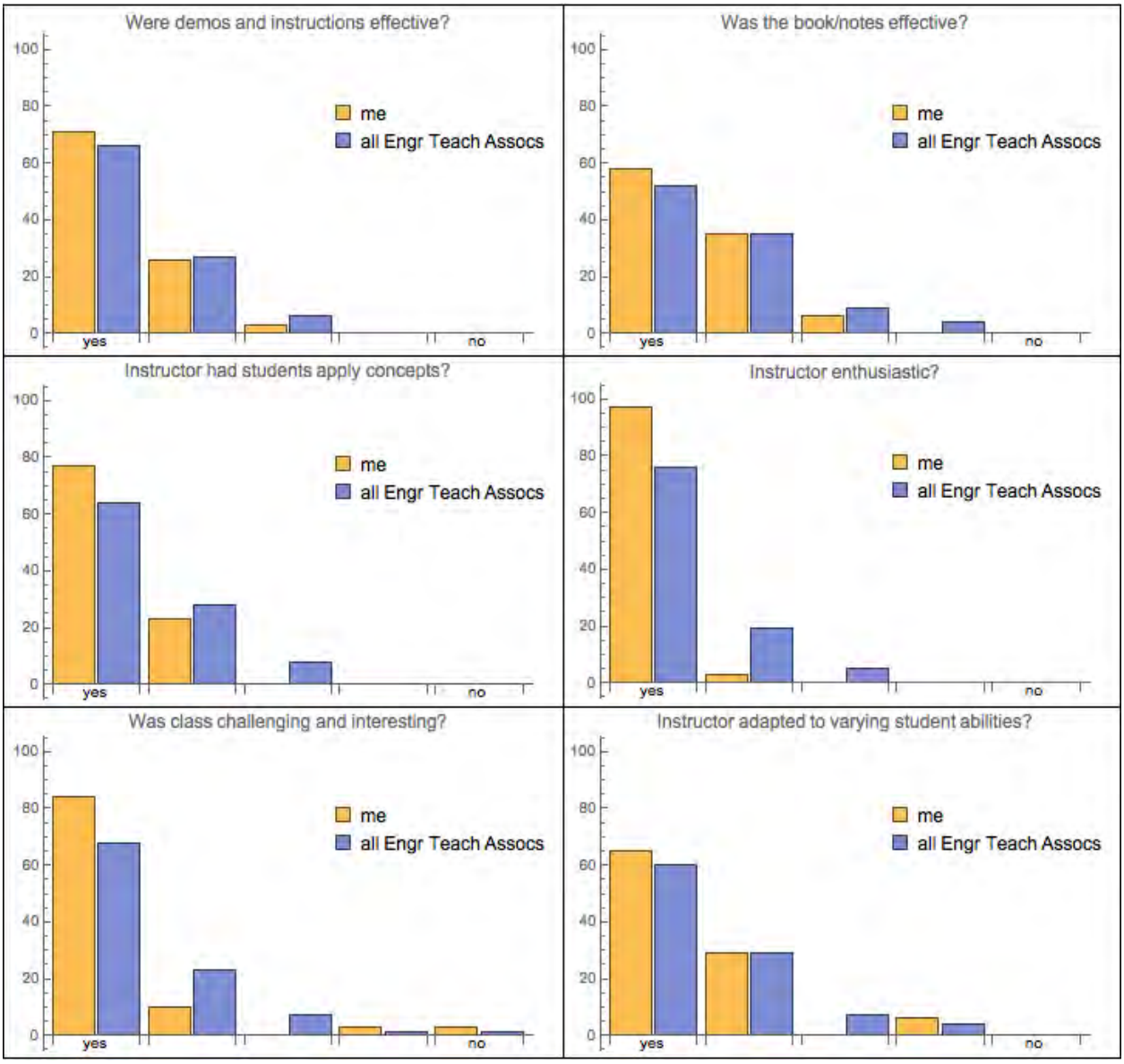

Adjusts his method of explanation to the students' level of understanding?
me
all Engr Teach Assists
yes
no
Available outside class hours?
me
all Engr Teach Assists
no
yes/dunno
Are you afraid to ask questions for fear of imposing on his time?
me
all Engr Teach Assists
yes
no
Do you feel you must have a specific question before talking to him?
me
all Engr Teach Assists
yes
no
How effective was his section in helping you make connections with the lecture?
me
all Engr Teach Assists
great
poor
How much did the discussion session contribute to the overall value of the course?
me
all Engr Teach Assists
a lot
a little
How useful was the discussion session for aiding your understanding of the material or your ability to apply it?
me
all Engr Teach Assists
great
poor
TA's overall performance and helping you master the course.
me
all Engr Teach Assists
A
F

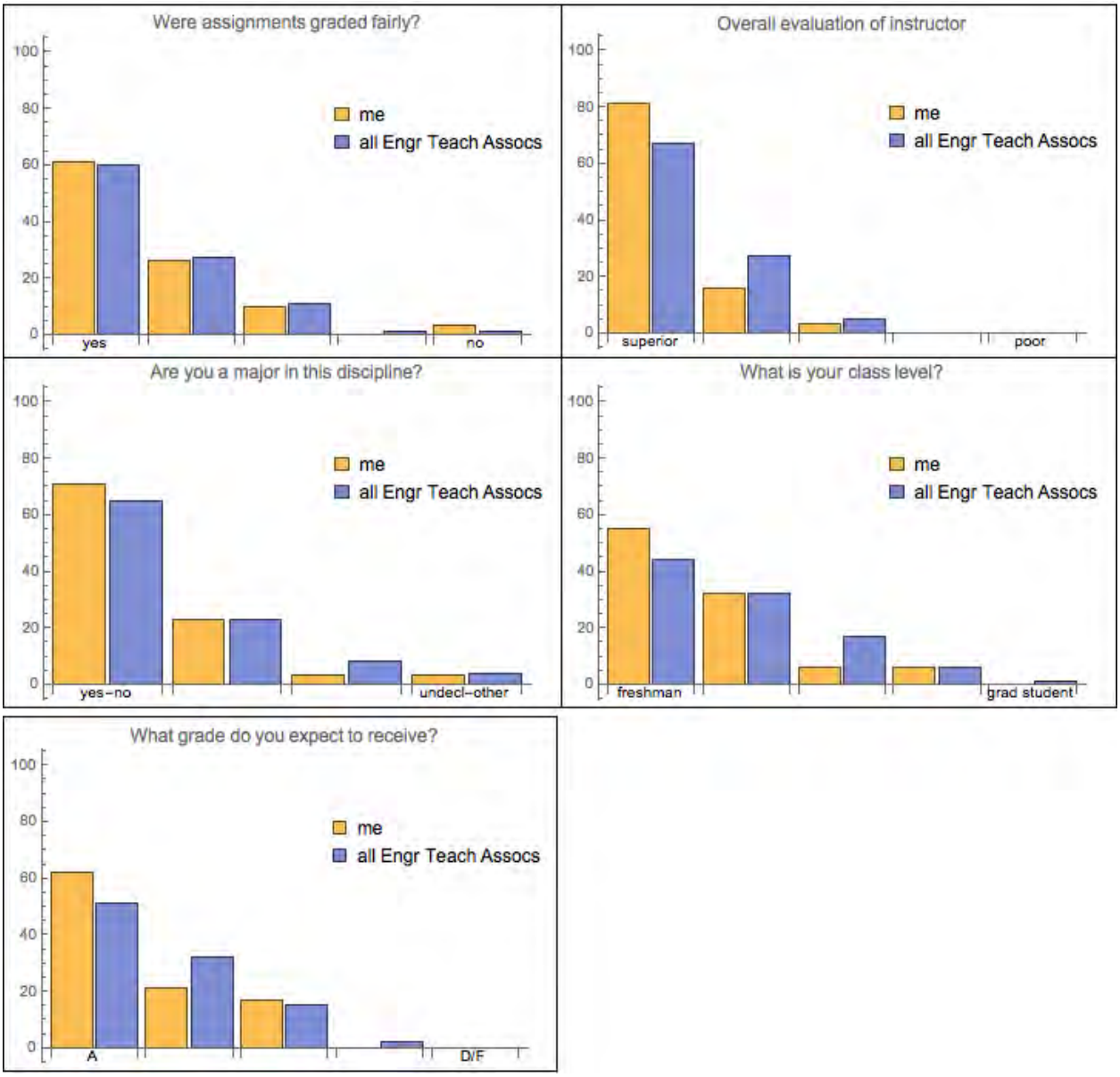
CHAPTER

TWELVE

APPENDIX 5: ESCI TEACHING EVALUATION DATA (INSTRUCTOR, ENGR 3)



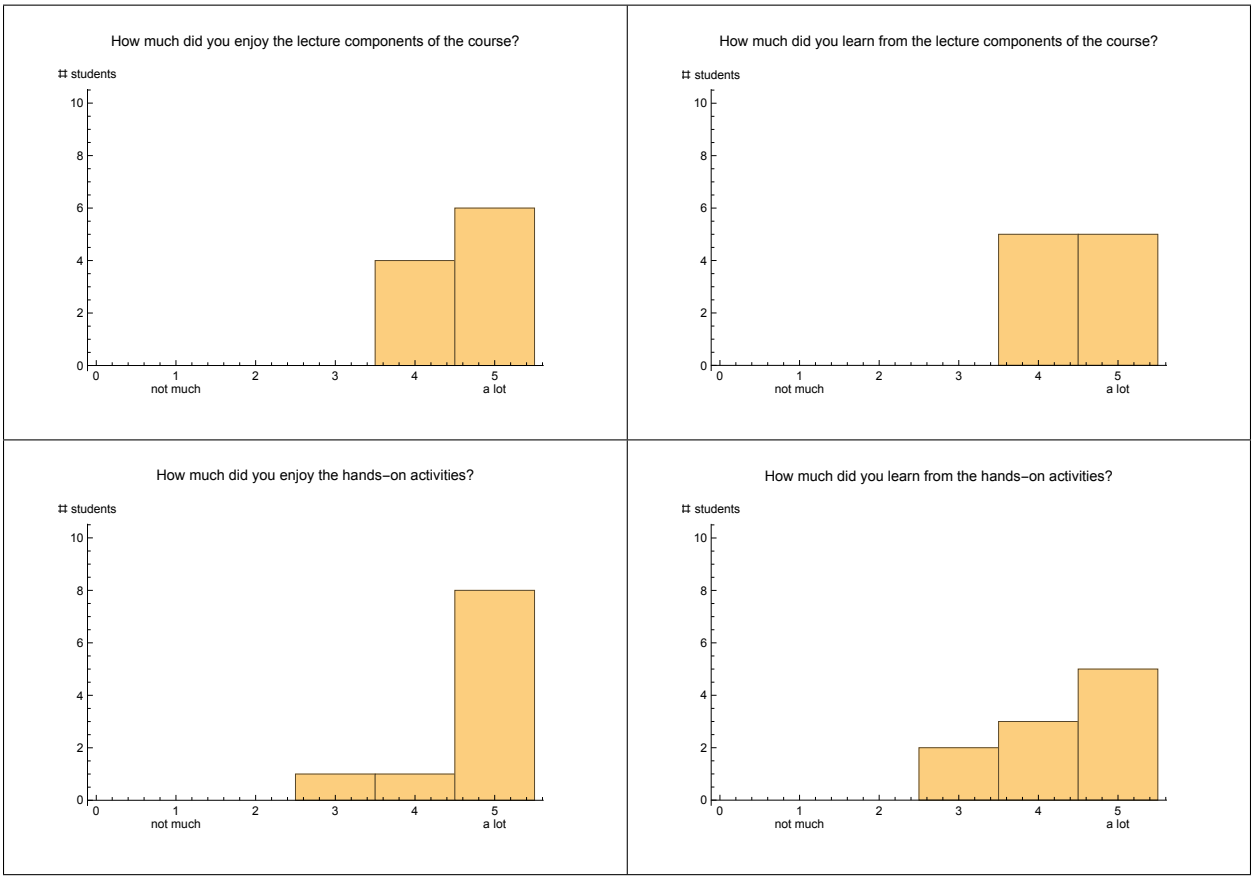

Were demos and instructions effective?
Was the book/notes effective?
Instructor had students apply concepts?
Instructor enthusiastic?
Was class challenging and interesting?
Instructor adapted to varying student abilities?
me
all Engr Teach Assocs
yes
no
0
20
40
60
80
100


Were assignments graded fairly?
100
80
60
40
20
0
yes
no
me
all Engr Teach Assocs
Overall evaluation of instructor
superior
poor
Are you a major in this discipline?
yes–no
undecl–other
What is your class level?
freshman
grad student
What grade do you expect to receive?
A
D/F

CHAPTER

THIRTEEN

APPENDIX 6: SST STUDENT FEEDBACK



What did you like most about your SST experience?

- I was able to learn to program and use actual robots to do it.
- The application of coding and information on control theory, as well as the usement of technology in everyday life.
- The learning
- I learned more about science and met new people.
- Hands on programming and working with the robots.
- Everything it was fun.
- Getting to learn how to operate the robots.
- Meeting new people.
- Learning the concepts.
- I really liked the kind instructors and the great food offered after every class.

What was the most interesting topic you covered in class?

- Everything to be honest, converting bits and bytes, as well as programming on java.
- Control theory, java, realterm
- To me the most interesting topic was the converting of regular numbers into the same amount but in 0+1s
- The control theory and java.
- Actually programming the robots.
- Programming using Java.
- How to stabilize the beam at a 90 degree angle.
- Control.
- The most interesting topic we covered for me was schematics and how it can apply to a physical system.

If you were to take this class again, what suggestions would you have for improvement?

- Be able to have the classes longer.
- A hands on example by the teachers first.
- I don’t know.
- To use other programs beside JAVA.
- A little more time working with the robots.
- None, the time was not all too many so their was nothing that they could be done.
- Get an understanding of how the insides of a computer can translate code.
- I would just suggest a larger of classes of 8-10 weeks.

What did you like about the instructor’s teaching style?

- I liked how he was able to explain complicated concepts in a simple way.
- They explained everything clearly and to the point, they also used examples.

- They helped you and actually explained what you didn’t get in a friendly way.
- I liked that they would let us ask questions and did not make it so different.
- They both were very clear and conAnd were able to answer any questions we had.
- The relaxed teaching style.
- I like his power pointsw with exmaples to get a much easier explanation.
- Quick to the point but with some supplementary stuff thrown in.
- Don’t know exactly.
- They were both very real and upfront with us and always allowed us to have a variety of experience.

How could the instructor improve on their teaching?

- They did a great job teaching.
- Perfect/more hands on activities.
- I don’t know.
- The instructor did a good job.
- From what I experienced they both did very well.
- They did the best they could in the time provided.
- Going step by step on the programs for the laptops to help the student understand how to set up.
- Was very good so I don’t know what there is to be improved.
- I was just not sure how to organize or interpret the coding concepts well enough to implement them extensively.

CHAPTER

FOURTEEN

APPENDIX 7: BLOOM'S TAXOMONY OF EDUCATIONAL OBJECTIVES

ED 253D introduced me to a theory that strongly resonates with my view of learning: *Bloom's Taxonomy of Educational Objectives*. Depicted *here*, it is a model for how a student's mastery of a subject improves with practice. Bloom proposed that a student's proficiency develops in a fixed sequence of cognitive stages — from "Lower-Order Thinking Skills" to "Higher-Order Thinking Skills" — as follows:

1. **Remembering**. First, students must be introduced to the new knowledge, and they must remember it.
    - Here is the mathematical recipe for how matrix multiplication works.
2. **Understanding**. Next, students must get a sense of the new material: Explain it plainly and relate it to other familiar concepts.
    - The formula for matrix multiplication amounts to taking rows of this matrix and dotting them with columns of that matrix. It is just a way of doing arithmetic in batches. Here are some examples of matrix multiplication...
3. **Applying**. Students practice the knowledge and get a feel for using it.
    - Now you try. Multiply these two numeric matrices. Next, try multiplying these symbolic matrices.
4. **Analyzing**. Now that students can "blindly" apply this new knowledge, they need to integrate it with their existing knowledge.
    - Matrix multiplication is a generalization of normal multiplication of real numbers. Many of the rules for algebra on the real numbers work for matrices as well. For example, multiplication distributes over addition.
    - You can use matrix multiplication to do certain types of transformations on vectors — rotate them and flip them.
5. **Evaluating**. Students explore the material more deeply, finding extensions and shortcomings of it.
    - Unlike a real number, a matrix is not always have a multiplicative inverse. What properties does a matrix need to have in order to have an inverse?
    - A matrix can rotate a vector, but not translate it. How would I have to augment a matrix if I wanted to rotate *and* translate it?
6. **Creating**. Students invent their own way to apply this knowledge.
    - Here are some example areas where matrices arise: ... Now, invent of a real-world math problem that can be solved using matrices and matrix inversion.

CHAPTER

FIFTEEN

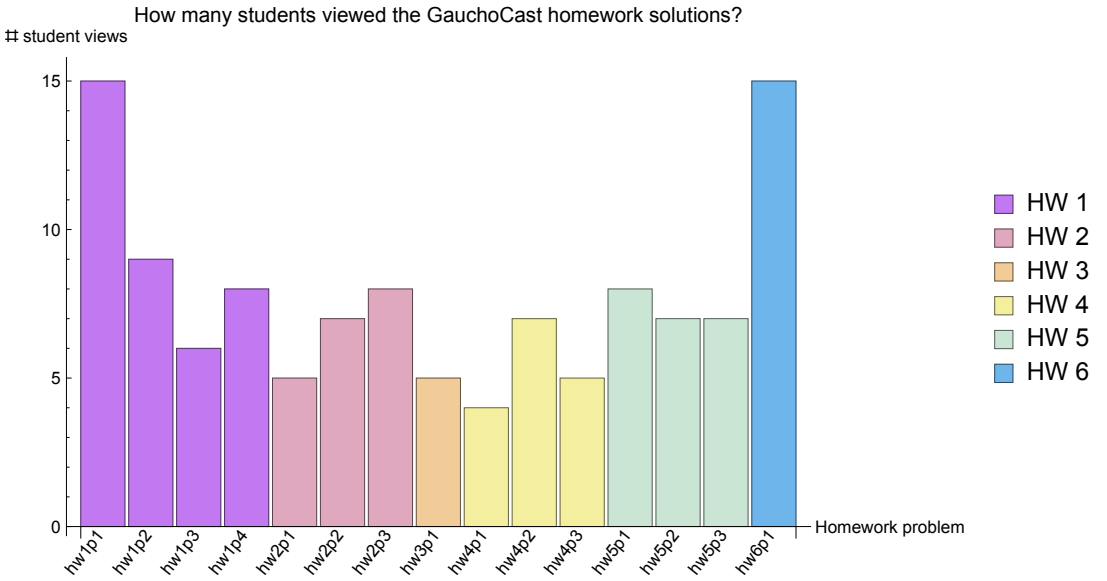
APPENDIX 8: GAUCHOCAST VIEWING ANALYSIS

In this appendix we explore whether my GauchoCast video solutions from my Summer 2015 ENGR 3 course were better than just posting PDFs of the homework solutions.

GauchoCast offers richer usage statistics than mere PDFs. With PDF solutions, an instructor can determine only that a student downloaded the PDF, not whether the student actually opened the PDF and studied it. In contrast, GauchoCast records which students viewed the videos, when, and for how long. This gave us much better insight into how many students accessed our solutions.

How many students watched the homework solution videos?

This plot describes how many unique students viewed each of the 15 GauchoCast videos (one per homework problem):



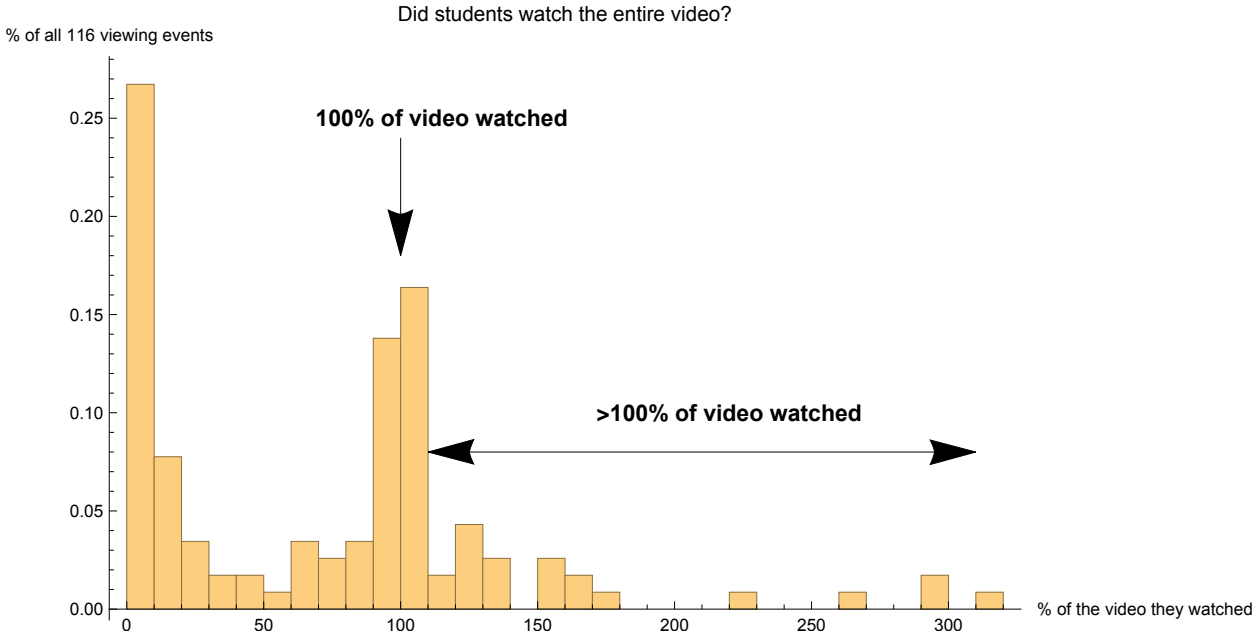
On average, 7.7 students viewed each video (23% of the class). Many students watched the first video because it was novel, and many watched the last video because it was released the night before the final exam. But the majority of the videos were largely ignored by most of the class.

When a student watched a video, how long did he or she watch it?

To answer this question, we tabulated the fraction of each GauchoCast video (each representing one homework problem's solution) that each student watched. Specifically, for each video and each student who watched some of it, we tabulated how many seconds the student watched the video, expressed as a percentage of the video's length. For example, if a student watched 4 minutes of the 5-minute video for homework 1 problem 3, we call this an *80% viewing event*. It still counts as an 80% viewing event if the student watches two 2-minute chunks of the video.

Note that >100% viewing events are possible. This occurs if a student watches the video repeatedly, say, once after it is released and again in preparation for the exam.

Ideally, each of the 33 students would watch each of the 15 videos to completion, resulting in 495 "100% viewing events". In reality, there were only 116 viewing events, whose viewing-percentages deviate wildly from 100%. The viewing-percentages of the 116 viewing events are shown in the following histogram:



From this graph, we observe:

- The 26% hump on the left corresponds to <10% viewing events, i.e., 31 of the 116 viewing events correspond to a student watching less than 10% of the video. This may correspond to students pausing the video to examine the computer code, but not bothering to listen to the code's explanation.
- The hump at x=100 corresponds to viewing events where the students watched 100% of the video (as intended).
- The scattered bumps with x>100 correspond to viewing events where students re-watched videos many times.

Some relevant statistics of this histogram are summarized in the following table:

Percentage of student-views	Percentage *p* of video watched
18%	p ≥ 110%
30%	110% > p ≥ 90%
25%	90% > p ≥ 10%
27%	10% > p

Practically speaking, this means that when a student views a video:

- A sixth of the time, they'll watch it multiple times throughout the course (>110%)

- A third of the time, they'll watch it as intended (~100%)
- A quarter of the time, they'll watch some of it (10%-90%)
- A quarter of the time, they watch practically none of it (<10%)

In summary, the GauchoCast homework solutions were utilized by roughly a quarter of the class, and of those, about half watched the videos to completion. Their viewing habits exhibited considerable variability. The most important question — *did the students benefit more than from traditional PDF solutions?* — remains unanswered due to difficulty in assessing how many students study PDF solutions and to what degree. But at least these results establish a baseline for further investigation in subsequent courses.