Survey Results

We sent out a 5-question survey to around 10 high school physics teachers that we knew. Only 3 teachers responded, but we integrated their responses into the design of our materials. We have summarized the survey data below:

How much computation have you been exposed to?
Teachers had varied experiences. One teacher had limited experience with a variety of languages (C++, python, MATLAB, and java), another teacher worked as an undergrad with a research group using Fortran to model general relativity, and the last teacher used computer simulations but never interacted with the code.

Do you currently use computational methods in physics instruction, and if so, how?
Two teachers reported that their students use excel to compare data to a model, and one teacher reported using PhET simulations in the classroom.

In what ways might you integrate computational methods into your physics curriculum? Please be specific.
One teacher reported that she would have students create simulations for kinematic phenomena as well as have them model things at large and small length scales (microscopic, molecular forces, orbital mechanics, etc.). Another teacher said that he would use spreadsheets to work through problems with air resistance, and commented that it might be a good place to have students think about loops. The last teacher did not have ideas and expressed concern over integrating something new into an “already-packed curriculum” and said that he would need to be convinced that there is merit to implementing computation into the curriculum before attempting it.

Is there anything limiting your ability to integrate computational methods into your curriculum (resistance from your school, students' experience, funding, etc.)?
Two teachers reported that computer access is limited (in one case, there is only one set of classroom computers for an entire department of 16 teachers. In the other case, there are 7 computers for a class of 24 students). All three teachers also reported that timing is an issue, both in implementation and in planning. One teacher cited AP curriculum constraints, and one teacher cited her lack of experience in computation as a limiting factor.

Anything else you want to add? We'd love to hear about it!
One teacher (who was a Noyce fellow at Tufts a few years ago) requested that you offer your course in the summer so that she could take it(!). If not, she asked if you could email her some of your lecture notes and/or problem sets (I have not responded to her about this, Tim, so let me know what you think!). Another teacher said that he did not envision incorporating programming into a first year AP physics class but could easily see incorporating it into a second year AP physics class or (interestingly enough) a non-AP class.

Design Choices

A few teachers (who did not fill out the survey) expressed confusion over what we even meant by computation and computational methods, which pushed us to think about how teachers who have never been exposed to computation might utilize these materials. I (Jen) had no computational experience when starting this class, but now that I have a feeling for what kind of a thing computation is and does, I can think of many ways that computation could be useful for introductory physics learning. Similarly, we felt if teachers understood the basic affordances of computation, they would be able to use it in their classrooms in ways that extend beyond the context of these particular materials. In order to acquaint teachers with the basic affordances of computation, we decided to create a document that explains what computation is and how it can be useful.

To address the concern that teachers do not have time to implement something new into an "already-packed curriculum," we cited areas from the NGSS and the *Framework for K-12 Science Education* that focus new attention on computational competencies. We would have liked to do a more in-depth literature review, but our time was limited.

Due to the many doubts that the teachers expressed in these surveys about students' ability to learn programming, we decided to create two different instructional materials. The first is a mathematica notebook simulating air resistance in which students could (1) toggle lots of variables and see how the y vs. x, y vs. t, vy vs. t, and ay vs. t graphs change and (2) upload and analyze video of themselves dropping things to see how well real-world data fit the model. The second is a "tutorialized" python script that scaffolds students' programming experience to create their own simulation using visual python.

The Python template is modeled after computer science assignments we have encountered in CS courses at Tufts. It gives students detailed, step-by-step instruction on what each function needs to perform as well as an API to introduce them to the syntax and properties of vPython. It also contains questions about why the simulation runs the way it does, what effect the various parameters have on the simulation, and what features can be added that enhance the physicalness of the system (such a resistance coefficient on the ground and different calculations of drag). The goal of the template is to be accessible to new programmers but challenging enough for enthusiastic students.

Iterations

vPython

- First made fully functioning simulation
- Split the simulation into two programs, one main simulator that need not be edited and a list of modular functions to handle the calculations
- Kept one fully implemented version for reference and created a student version with empty function definitions and detailed instructions
- Gave to Jen (our test-student!) to interact with and get feedback.

Mathematica

- First created manipulate to show y vs. x, vy vs. t, and ay vs. t graphs with toggleable parameters (drag coefficient, object mass, gravity, vo, starting angle). Time can also be toggled. The first version had every graph as a separate manipulate.
- Put them all on the same manipulate, so the graphs would all display the same parameter values.
- We ran it on the Raspberry pi and it ran extremely slow, so we changed some of the code to run faster. We changed it so that the only automatically updating parameter was time, and the other parameters, once set, did not get recalculated at every changing time step.
- It was still slow on the raspberry pi, so we decided to scrap that, and return to automatic updating of all parameters so that students could see how the graphs change when changing other parameters. We thought this was interesting, and provided some “thought questions” that were easier to answer with automatic updating for all parameters.
- We added the code Tim wrote for video analysis so that students could take videos of dropping balls and compare slices of those videos with the y vs. t graph. We wanted to try and overlay it but couldn’t get it to sit in the right place on top of the resliced images.

Technological Limitations

- Mathematica ran extremely slow on the raspberry pi. Even after making our Mathematica code more efficient.
- vPython had some problems defining certain objects on the raspberry pi. We tried to update the system and upgrade to Python 2.7.9, which is required to run vPython correctly, but it took a very long time and at the end aborted the installation for unknown reasons. We were unable to run our vPython code on the Raspberry pi successfully.
- We had trouble finding ways to connect the raspberry pi up to a monitor (it only came with an HDMI cord, which does not connect to most computer monitors that we tried). We also had trouble setting up the wifi card and ended up having to connect it directly through an ethernet cord. Since any teacher using a raspberry Pi would also need to purchase monitors, keyboards, and mouses, have an internet hookup and since we couldn’t get the Mathematica visualization code to work (which is the main reason to invest in a Raspberry pi), we would recommend teachers purchase a set of cheap laptop computers (possibly Chromebooks) and run python through those computers.

Team Expertise

- It was extremely helpful to have some members on the team proficient in programming (Raewyn & John), and some pretty unfamiliar with programming (Jen) so that we could see the python script through the eyes of a (semi) novice.
- It was also helpful to have someone with expertise in education.