Incorporating Computation Into an Introductory Course at a Liberal Arts University

Incorporating computational elements into classes at a small liberal arts college can be a challenging endeavor. In many introductory level courses, the class is made up of students from all different quantitative abilities and interests. In my Introduction Engineering class, for example, I have had everyone from physics majors to studio arts majors. But that’s not to say an artist or an education student or a biologist have no need for computational curriculum in the course. On the contrary, these students bring with them a different perspective and, though they may never find themselves turning to Matlab or any other computational platform later in their careers, the type of ordered thinking and problem solving inherent in computer programming will benefit them in many different areas.

Students, especially students who do not consider themselves as skilled in quantitative fields, quickly become overwhelmed when presented with a computation task. Often they approach me a few minutes after the assignment has been presented and tell me nervously in a whisper “I’m a [insert nontechnical field here] major.” Fortunately for them, the value in a liberal arts education is to stretch one’s self beyond the field in which they are most comfortable! Some careful attention to scaffolding computational work is pedagogically effective and also reassures students that you are not out to make them feel foolish. Building computational work into the curriculum as early and at as low a level as possible allows those students unfamiliar with the technology time to build their skills and confidence. For example, the last ten minutes of class in the first week of the semester could be used to allow the students time to practice solving some simple arithmetic problems in Matlab using both the command line and as a script. It is important to not overwhelm them with both a new quantitative topic in addition to a new computational topic at the same time in introductory courses. If the curriculum is novel, the computational element should be familiar and vice-versa. As the course material becomes more complicated, the required Matlab code can grow in complexity alongside. Class time should be dedicated to practicing more challenging programmatic concepts, like loops, before they are required for an assignment.

In a different course, Engineering Statics, one of the student learning outcomes is to become proficient at using computational tools to solve problems presented in the class. This includes choosing the appropriate computational tool in addition to developing a robust solution. Students soon realize that solving complex truss analysis problems is easier and more interesting when they first develop a Matlab script to represent the system of equations governing the truss. Once they develop a model, it is easier for them to vary things like the geometry of the beams, for example, to maximize the strength of the truss. By the end of the semester, the goal is for students to become independent, creative, and confident when applying their computational skills.

Both technical and non-technical students benefit from learning Matlab in these courses. Non-technical students gain the sense of accomplishment which comes from mastering a skill they originally believed was beyond their ability. And our physics students learn to incorporate a valuable analysis tool that they will continue to use whether they choose to pursue a graduate degree or work in industry.