The Journal of Computing Sciences in Colleges, vol. 25(6), June 2010, pp. 60-65.

AN INTRODUCTORY COMPUTATIONAL COURSE FOR SCIENCE STUDENTS

Sohie Lee and Ellen Hildreth
Computer Science Department
Wellesley College

ABSTRACT

This paper describes an introductory CS course, in a liberal arts setting, designed for science students to learn programming using MATLAB. This course differs from many introductory CS classes in that 1) students are not expected to continue taking more CS classes and 2) MATLAB is used in a general computational context rather than in an engineering setting. Students learn to write software to solve problems, visualize and analyze data, perform computer simulations, and implement and test computational models that arise in a wide range of scientific disciplines. The course culminates with an individualized final programming project in which students apply their MATLAB skills in their particular area of scientific interest.

INTRODUCTION

Science requires computation. There are data to be analyzed, changes to be measured, and predictions to be made. The manner in which established scientists learn about computing, however, is typically haphazard [4]. At liberal arts colleges, from which graduates go on to earn PhDs in the sciences at a higher rate than other undergraduate institutions [1,2], training in computing is essential. We designed a CS0 course to teach science students about computation using MATLAB.

At the time the course was developed, our CS department offered two introductory courses: a Java based course for CS majors, and a course for non-CS majors on web design using HTML and JavaScript. Our goal was to create a CS0 course aimed at science and social science majors who do not intend to enroll in further CS courses. Science and social science students need to learn computation, not as computer scientists per se, but rather as scientists who need to compute.

MATLAB was chosen because it is a powerful and widely used technical computing environment with advanced graphics, visualization and analysis tools, and a rich high-level programming language. MATLAB is widely available on campus. One advantage of MATLAB is that students can quickly generate meaningful programs and results, an important factor in introductory programming classes. In our course, MATLAB is a tool for teaching computation, much in the vein of Kaplan [3,4], as opposed to a CS0 course targeting engineering students [6].

CURRICULUM

Our goal was to provide a solid foundation of basic programming concepts, with an emphasis on computing that is particularly useful in the sciences. Accordingly, our topics fall into three main categories: 1) Data representation and

organization, 2) Program structure and flow and 3) Data analysis and modeling. In data representation, we cover the various ways that scientific data, both numerical and textual, can be stored in MATLAB. This includes variables, matrices, strings, structures and cell arrays. For program structure and flow, we teach the fundamentals of writing individual functions using conditionals and iteration (including recursion), as well as the higher level skill of how to design a large program using multiple functions and subfunctions. The design process emphasizes the important ideas of abstraction and modularity. We also teach how to create graphical user interfaces (GUIs), and merge the underpinnings of a program with the user interface. For the third category of data analysis and modeling, we cover reading external files of many types, writing files in a useful format, visualizing 2D and 3D data, and selecting and sorting data. We also introduce the powerful MATLAB toolboxes. Specifically, our students learn about the Curve Fitting, Image Processing and Statistics Toolboxes, as well as the Interactive Plotting Tool.

The course meets three times per week for 13 weeks. Each week includes two 70-minute lectures and one 110-minute hands-on laboratory. During lecture, students often engage in group problem-solving. The laboratory revolves around generating MATLAB code that instantiates concepts from the previous week's lectures. There are typically 8 homework assignments, each requiring 6-8 hours on average, that are completed outside of class. There are also two in-class examinations and a final project.

Our homework assignments have illustrated many different ways that MATLAB can facilitate problem-solving and computation. Students have written MATLAB programs to perform tasks often needed in the world of science. Sample tasks included: examining the frequency of occurrence of structures or events, e.g. identifying a foreign language given occurrences of the most common letters [5]; simulating dynamic processes, e.g. population growth or spread of disease in a population; performing statistical analyses, e.g. removing outliers in a data set, or extrapolating from a data set to make predictions; performing graphical analyses of data, e.g. energy production and consumption data, and supply and demand data; creating synthetic images, e.g. visual illusions and recursive pictures; image processing, e.g. counting grains of rice using luminance data and analyzing the spectral composition of astronomy images; processing text, e.g. translating nucleotide sequences to amino acids [3] and selecting or sorting numerical or textual information in a database, e.g. manipulating data in a bird species database. Our choice of programming tasks emphasized general problem-solving skills that are relevant in the sciences.

We wanted to build a level of comfort and expertise with MATLAB within a single course so that after completing the course, students would naturally turn to MATLAB (vs. Excel, Maple, and other statistical packages) for routine tasks like plotting and analyzing data as well as for more complex programming needs. MATLAB has the tools for these routine tasks seamlessly integrated into a general programming environment. This provides much greater flexibility for students to

customize their data analysis and visualization tasks, and to build a program that can integrate many analysis steps efficiently, and a GUI that allows the user to work with the data interactively. With this in mind, our course culminates in a final project where each student builds a substantial MATLAB program from scratch, drawing upon the wide array of computational resources that it offers.

A SAMPLE ASSIGNMENT: MRI DATA DISPLAY

Our students wrote an interactive program to display MRI brain slices, using data that comes with MATLAB. The assignment required reading in MRI brain data from a file and then a) offering display options in various orientations (sagittal, coronal, horizontal and vertical) and b) showing a movie of brain slices. Students initially wrote a text-based interface that allowed the user to select which orientation and which slice to view (see Fig 1). Several weeks later in the semester, the students revisited this program, and created a GUI with menus and buttons (see Fig 2) that enabled colorful three-dimensional displays of the same data.

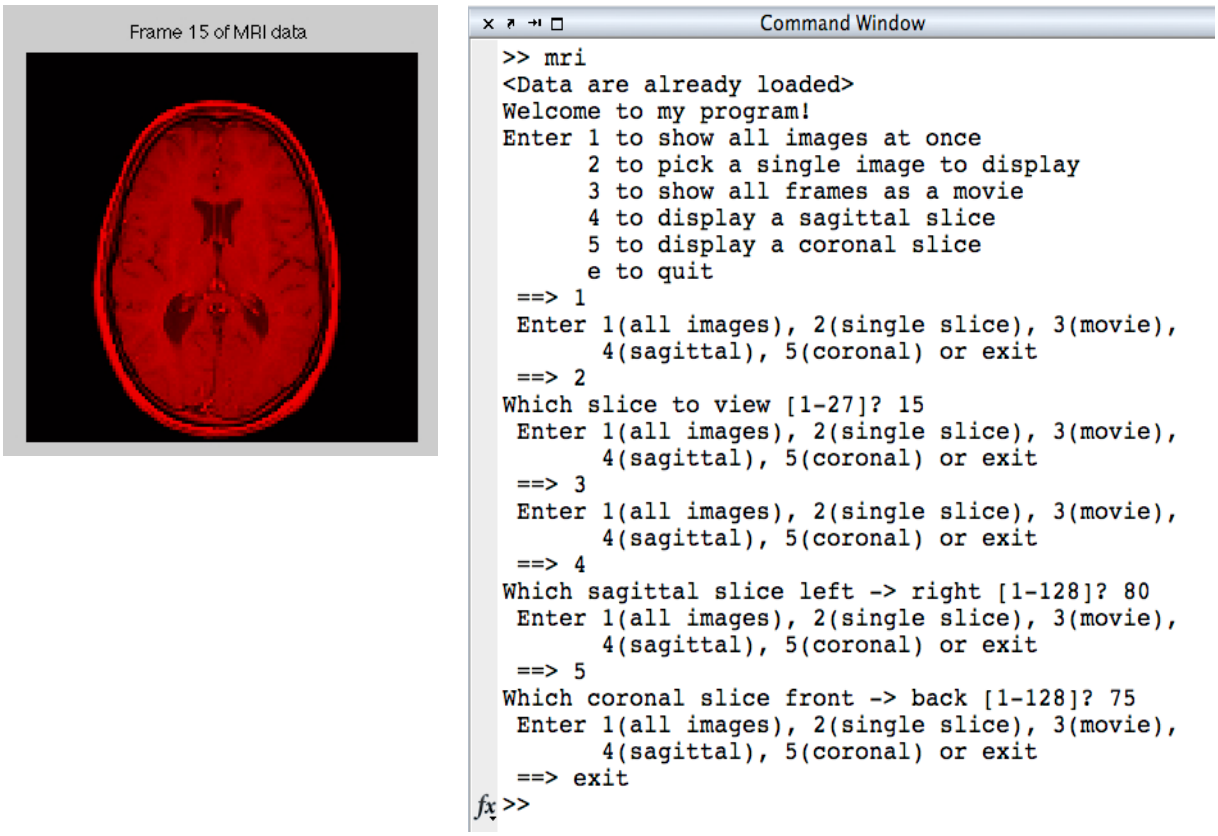

Figure 1. Screenshot of sample brain slice (left) and text user interface (right)

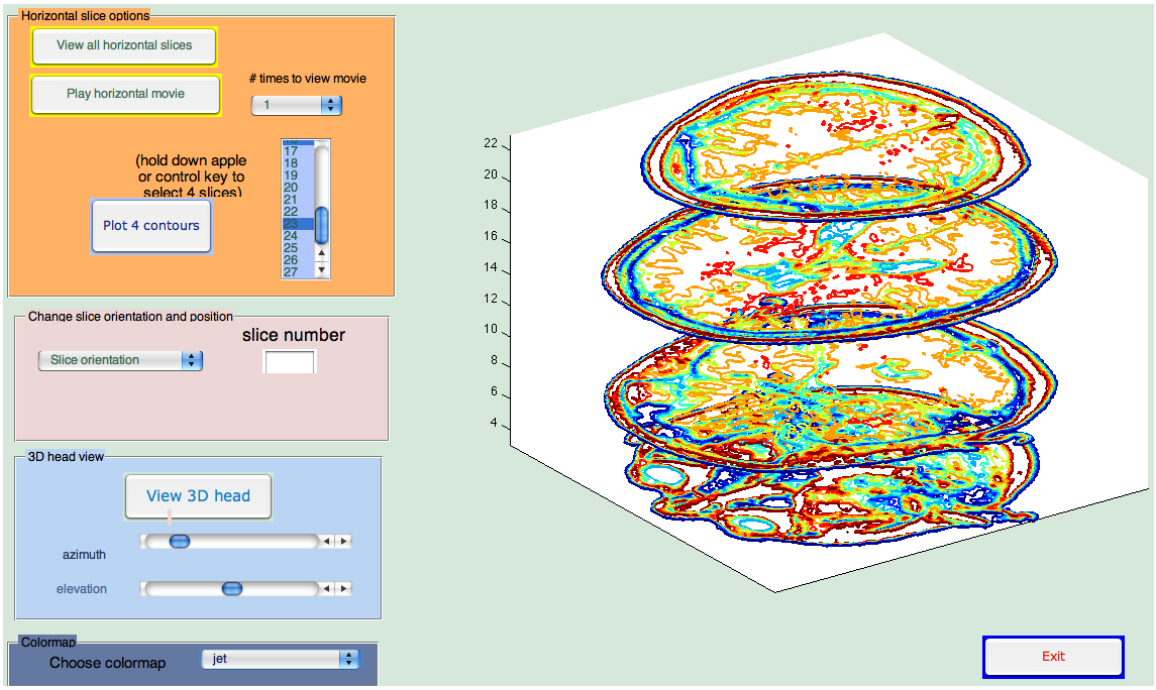

Figure 2. Screenshot of 3D MRI GUI

The development of the MRI brain slice assignment in two parts allowed us to emphasize some key computational ideas: 1) the ability to read an external file and display and morph its contents, 2) the division of a large program into smaller functions, and 3) the separation of the program's function from the program's user interface. The students were introduced to the MRI data early in the semester, before GUIs were taught. The second MRI assignment used the same data, yet the students wrote a completely different and more complex graphical interface that displayed the brain images in three dimensions. We emphasized the separation between the computation that a program performs and its user interface. One goal of this assignment was to provide a model by which the students understand how to decompose a problem into smaller units, and then build a new, more powerful and flexible user interface for that program.

FINAL PROJECT

The final project provides students with an opportunity to integrate and reinforce the many skills learned throughout the course. Each student designs a large project from scratch, typically 2-3 times larger than a homework assignment. Most students design and implement the program largely independently. This process has many benefits for our students. Creating their own project builds students' confidence and develops their self-reliance and resourcefulness. While working on their projects, students sometimes learn new MATLAB that is particularly relevant for their project. Independent programming on a self-chosen topic also helps students stay highly motivated and appreciate the relevance to their field of interest. Many of our course's final projects have supported research

projects with other faculty, providing our students with a source of pride when demonstrating their project to their research advisors. Faculty in other departments have directed their research students to our course, as MATLAB is a valuable tool in their research. As our final project requires a series of written reports as well as presentations, this strengthens students' writing and public-speaking skills. Finally, as a long-term assignment spread out over 6 weeks, the final project requires strong time-management and resource allocation skills.

Students chose final project topics based on their own interests. The final project consisted of four phases: 1) a general text description of the project with sketches, including lists of each MATLAB function or script; 2) a detailed code skeleton with roughly one-third of the code written; 3) an informal presentation of the project in progress during class; 4) a final project demonstration with the course instructors, submission of code and documentation. Table I lists a sample of the wide variety of final projects. Virtually all projects implement a graphical user interface; many projects read input data files and output results files.

Table I. Final projects in Computation for the Sciences

Simulation	Highway traffic jam formation model, airplane flight simulation, mass-spring system, 3D trajectory of ball under gravitational and wind forces
Educational Tools	Interactive periodic table, visualization of electrical and potential fields around charge distributions, economic forecasting, analysis of bonding in chemical structures, simulation tool for teaching game theory
Research Assistance	Data coding and analysis for motor learning study, data tabulation from internet search of gene and function correlations, analysis of MRI data from the College MRI facility, analysis of Maine coast tidal data, analysis of eye tracker data from perceptual experiments, analysis of NMR spectral data to determine molecular structures, analysis of peak absorbances and phase changes from chemical wavelength and absorbance data, creation of experimental color perception, analysis of eye alignment in historical images of visual artists
Fun & Games	Mastermind, Snake, Frogger
Miscellaneous	Travel planner, health risk evaluation, decryption and encryption

CONCLUSIONS

Computation for the Sciences has been taught 5 times in 3 years to students whose majors represent many diverse fields (Biology, Chemistry, Physics, Astronomy, Economics, Computer Science, Neuroscience, Cognitive Science and Psychology). The students ranged from first-years to seniors. Within the broader

curriculum at our college, this course satisfies a distribution requirement in the area of mathematical modeling and problem solving. Although this was not the intent of this course, several students who took Computation for the Sciences in MATLAB as their first CS course subsequently declared CS majors or minors. The foundations of programming were sufficiently covered such that students have successfully transitioned into more advanced CS classes after our course. Feedback from the graduates of our course has been very positive. In the students' anonymous end-of-semester evaluations, 92% indicated that they would recommend our course to fellow students (78% indicated they would "strongly recommend" the course). One student summarized her course experience (Spring 2009) as follows: "I wanted to learn how to use a tool that would help me become a better scientist, and maybe learn a bit about computer programming. I'd wholeheartedly recommend this course to students who want to learn a programming language that is useful for the sciences." We have also received positive feedback from faculty in other departments whose students have taken our course.

Perhaps the most compelling examples are those students who, after taking our course, continue using MATLAB for computation in their independent research projects, or in their field of choice. At our college, MATLAB is widely used in teaching and research in disciplines such as Neuroscience and Physics. Some students in these and other scientific fields have made it clear that knowledge of MATLAB programming from our course facilitates advancement in their scientific careers. For example, one graduate of our course uses MATLAB extensively for her Geosciences thesis project, and is able to efficiently analyze and present her data to her advisor, who is not familiar with MATLAB. Other students are using MATLAB in their graduate or professional work, in areas such as physics and medical imaging.

ONLINE COURSE MATERIALS

http://cs.wellesley.edu/~cs112/courseMaterials

REFERENCES

[1] Cech, T.R., *Science at liberal arts colleges: a better education?* in S. Koblik and S.R. Graubard, eds, Distinctively American: the Residential Liberal Arts Colleges, Transaction Publishers, 2003.

[2] Freeman, R.B., Jin, E., Shen, C-Y., Where do new US-trained science-engineering PhDs come from?, *National Bureau of Economics*, June 2004.

[3] Kaplan, D., *Introduction to Scientific Programming and Computation*, Pacific Grove, CA: Brooks/Cole, 2003.

[4] Kaplan, D., Teaching computation to undergraduate scientists, *Association for Computing Machinery*, 36, (1), 358-362, 2004.

[5] Kaplan, D., Scientific Programming Resources for Instructors, www.macalester.edu/~kaplan/ScientificProgramming/ForInstructors.html, December 2004.

[6] Tew, A. E., McCracken, W. M., Guzdial, M., Impact of alternative introductory courses on programming concept understanding, *International Computing Education* , 25-35, October 2005.