2006-1149: TEACHING THE INTRODUCTORY COMPUTER-PROGRAMMING COURSE FOR ENGINEERS USING MATLAB AND SOME EXPOSURE TO C

Asad Azemi, Pennsylvania State University

Asad Azemi is an associate professor of Engineering at Penn State University. He has received his B.S. degree from UCLA in 1982, M.S. degree from Loyola Marymount University in 1985, and Ph.D. degree from University of Arkansas in 1991. His professional interests are in nonlinear stochastic systems, control systems, signal estimation, bio-computing, and use of computers in undergraduate and graduate education.

Laura Pauley, Pennsylvania State University

Laura Pauley is a Professor of Mechanical Engineering and the Arthur L. Glenn Professor of Engineering Education at Penn State University, University Park. Since 2000, she has also served as the Professor-in-Charge of Undergraduate Programs in Mechanical and Nuclear Engineering. Dr. Pauley teaches courses in the thermal sciences and conducts research in computational fluid mechanics. She received her Ph.D. from Stanford University in 1988.

Teaching the introductory computer programming course for engineers using Matlab and some exposure to C

Abstract

The introductory computer programming course for engineers is usually taught using the C++ programming language. This work describes our current effort, as a pilot project, which can be used in an evaluation process by those departments that would like to substitute Matlab for C++. Those who would like to continue the current practice, but are looking for more challenging problems or projects involving Matlab can also use the project outcome. The main reason behind switching to Matlab from C++ is the fact that many engineering faculty at Penn State, in various departments, have recognized that the current courses teaching programming skills using C++ are not fully utilized in later required courses in the curriculum. Increasingly in undergraduate courses in various engineering disciplines, Matlab is being used for problem solving. Robotics projects, which include using Handy Board and Interactive C programming, were added to illustrate an application of programming which students can relate to and enjoy, as well as helping them to improve their software and algorithm design skills for real-time applications. A discussion of the advantages and disadvantages of conducting a computer programming course in this format, including three different course formats that we have used are included.

I. Introduction

Computer programming has been part of the engineering curriculum since the dawn of the computer age. The course is typically taught during the freshman or first semester of the sophomore year to ensure that students have sufficient programming background for solving problems in engineering courses. Although the assignments usually require some mathematical and/or basic physics/engineering background, the course is focused around programming concepts. Most universities use a "teach-a-language" approach in teaching this course, which means students work with a general purpose programming language (e.g., C++, Java, or FORTRAN) that is sufficiently flexible to build anything that needs to be built. This approach provides training in programming, but is so time consuming that there is little opportunity left to learn about computation.

II. Current practice

The introductory computer programming course for engineers, which was once dominated by FORTRAN for several decades, was gradually changed to C and later to C++ during the late 1980's and early 1990's. Currently most universities require only one computer-programming course for their engineering students (excluding computer engineering and computer science). This course is usually taught in C++ and in some cases in Java and FORTRAN. Until the fall 2004 semester at Penn State, we offered two versions of this course (C++ and FORTRAN), where the C++ version was required by the majority of the departments. The course was designed to cover fundamental concepts of programming (using C++ or FORTRAN), including introduction to computers and programming; data types, declaration and displays; assignment

and interactive input; selection; repetition; functions; arrays; strings; and input/output methods, with emphasis on engineering and advanced mathematics problems, and a brief introduction to Matlab for two weeks. As a pilot project, during fall 2004 two sections of the course, one at the University Park campus and one at the Delaware County Campus, were offered using Matlab. The section at the University Park campus was entirely focused on using Matlab and the one at the Delaware County Campus was designed around twelve weeks of Matlab and three weeks of C/C++, which was the reversal of our previous practice. Since, fall 2004 both campuses have continued offering these courses on regular basis. The University Park version of the course has been adapted as the recommended course for biomedical engineering students. The Delaware County Campus version of the course is up for evaluation before the end of summer 2006. The main reason for exposing students to C/C++ was the addition of robotics projects using Handy Board, which requires knowledge of Interactive C, a special version of C that was designed for running on a small 8-bit microprocessor-like the Handy Board's 68HC11. Robotics design projects were added to illustrate an application of programming that students can relate to and enjoy, as well as helping them to improve their software and algorithm design skills for real-time applications, and development of teamwork and collaborative skills, as well as improvement of communication skills, among other things. This work describes our experience at the Delaware County Campus of Penn State. It covers a brief discussion about the role and benefits of simulation software packages, our reasons for considering this change, three different course structures that have been used including topics covered and weekly schedules, sample programming assignments, challenges and problems associated with this approach, students' reaction, and finally our recommendations for those who are considering adapting a similar course.

III. The role and benefits of simulation software packages

The benefits of using simulation software packages, such as Matlab, in various engineering courses have long been realized by many educators, e.g. [1]-[8]. One of the main advantages of using these tools is the reinforcement of student understanding of theoretical principles by means of enhanced graphical aids. Simulation results can be used effectively in the classroom to emphasize the characteristics of devices and show the similarities and differences that exist between actual and theoretical characteristics. The simple graphical nature of the simulation outputs tends to help students understand the operation of mathematically complex system behaviors. Another equally important advantage is the preparation for analysis and the design of more complex systems than those that can be treated with pencil and paper, much like the ones students will see in industry. With these tools, instructors can assign complex design problems that otherwise would be unrealistic without the help of such software. This is a key advantage that helps students apply the theoretical principles learned in the classroom to the real world problems associated with following a design cycle through completion. The benefits of using these packages in a university setting are confirmed by the number of new textbooks and revisions of previously printed textbooks incorporating new exercises and problems based on these packages [9]-[19], among others.

IV. Why Change to Matlab?

As was mentioned, the introductory computer programming course for engineers at Penn State, until fall 2004, consisted of teaching programming skills using C++ or FORTRAN with a limited exposure to Matlab. It has been recognized by many engineering faculty at Penn State, in various departments, that teaching programming skills using C++ or FORTRAN was of limited value because these languages were rarely used again. Instead, increasingly in undergraduate courses in various engineering disciplines, they are using Matlab for problem solving. Moreover, Matlab and its toolboxes, due to ease of use and functionality, have become important tools for simulation based engineering research and even in some experimental setups. Therefore, we considered a pilot project to teach the programming course mainly using Matlab, with a limited exposure to C/C++. The limited exposure to C/C++ was added to accommodate the addition of robotics projects using Handy Board and to satisfy those departments that feel their students need to have a general understanding of one of these programming languages. Moreover, we are looking into the possibility of eliminating our linear algebra course and covering the bulk of the topics in the Matlab-based programming course. It should be mentioned there are few universities that either have converted their computer programming course to Matlab [20] or offer different sections of the course with Matlab and C++/Java options [21].

V. Course Structure

We started with an ambitious plan to teach the fundamental programming concepts, using Matlab, in addition to robotics-based programming projects, using Interactive C. The initial schedule (fall 2004) included twelve weeks of Matlab and three weeks of C/C++, which was later modified to four weeks of C/C++ so that we can cover more examples using functions. The C/C++ portion of the course included data types, declaration and display, assignment and interactive input, and functions. Interactive C specific syntaxes and functions were covered during the lab hours. The Matlab portion included Matlab editor/debugger, data types, declaration and displays, assignment and interactive input; collections, indexing, and selection, repetition, functions, plotting and model building, input/output methods, numbers and precision; topics in linear algebra, and an introduction to numerical calculus. Modular programming, using functions, and plotting were heavily emphasized throughout the course, see table 1. The second structure was used during summer 2005 where C/C++ and robotics portion were dropped and emphasis on linear algebra and numerical calculus were increased. It should be mentioned that the course was offered on an experimental 3-week format, see table 2. The third structure, used during the fall 2005 semester, was put together based on past course offering experiences. C exposure in this model was limited to the necessary information for learning Interactive C programming that was used for the assigned robotics projects. By doing so, we were able to dedicate more time on topics that our experiences indicated as crucial for improving student learning, such as modular programming in general, functions, role of functions and function formats, in particular. Table 3 presents the detailed daily schedule for this format. Three sample assignments from fall 2005 offering are also included in this paper.

VI. The Robotics Project

As was mentioned, robotics design projects were added to illustrate an application of programming that students can relate to and enjoy, as well as helping them to improve their software and algorithm design skills for real-time applications, and development of teamwork

and collaborative skills, as well as improvement of communication skills, among other things. In order to keep the emphasis on programming and not on robotics design, Carnegie Mellon's C-Based Robotics package was purchased. The package includes a step-by-step design of a tankbot and programming examples in Interactive C. Students were encouraged to use the tankbot in their projects. The final robotics project was our level-one "rescue mission" competition. Rescue mission competition is a more complex version of the original Trinity College "fire fighting" competition.

VII. Challenges/Problems and Recommendations

Matlab has been designed as a tool for simplifying technical calculations/simulations. It has not been designed to be a "programming language." With this in mind, some basic programming rules such as variable declarations have been relaxed. This constitutes a challenge in conducting a programming course using Matlab, so that it does not turn into a "teach-a-package" course. Therefore, if one looks at Matlab from a purely computer science point of view, it would not be the choice to teach programming courses. Another difficulty that we faced was the limited available resources in using Matlab as a programming language, versus the wide range of books that are published for use with general purpose programming languages. The course textbook [22] was chosen from a limited textbook collection [22]-[25] based on having more challenging engineering based problems. Unfortunately, looping and selection programming concepts were used in chapter 2 of the textbook, before these concepts were fully covered, for plotting and evaluating several complex mathematical problems. We consider this a shortfall of the textbook. Our recommendation for those who are considering using [22] is to skip those sections of chapter 2 that include such discussions/problems.

Based on our experience teaching two programming languages is useful only if the intent is to gain some exposure in one of the languages. As far as students' learning fundamentals of programming is concerned, the combination of more C++ and some exposure to Matlab worked better than more Matlab and some exposure to C++, The more Matlab and some C/C++ exposure can be improved if the C/C++ portion is focused on few specific projects and applications, than just general teaching of the language. In order to improve students' understanding of general programming principles, a greater emphasis on modular programming and functions are need during Matlab portion. An early exposure to functions is recommended. Robotics projects are useful in many respects but should be structured in such a way that follows the programming topics that are being discussed in the lectures and does not divert the focus of the course to an introduction to robotics course. In order to achieve this requirement, information regarding design, parts, and sensors should be readily available to students. With some modification, we intend to keep the robotics projects in the course. Based on our experience, if the course is solely focused on Matlab, with no C/C++, adding additional topics from linear algebra can justify the elimination of an additional course in this subject area.

VIII. Students' Reaction

Students' response to this approach was very positive. In spite of some difficulties with robotics projects, they all liked the idea of programming the robots. The only negative response was that they had to purchase Matlab!

Conclusion

In this paper, we have presented our experience in conducting an introductory programming course for engineers using Matlab, with addition of some robotics based programming projects. By taking a "teach-a-language" approach using Matlab, we can teach general purpose language skills and concepts and take advantage of its computational/graphical capabilities. This will give us the ability to discuss more advanced engineering/mathematical problems in a short period of time, which cannot be accomplished with general purpose programming languages. We should keep in mind that the main reason behind offering a programming course for engineering students is to help them with their future scientific computational tasks. We believe that Matlab can be used to serve this purpose. The fact that the majority of engineering jobs (excluding computer engineering which is not part of our discussion) do not require programming, but may require problem solving, should also justify the use of Matlab in engineering programming courses. Moreover, since Matlab is an integrated part of many advanced engineering courses and textbooks, an early exposure to this software is beneficial.

References

[1] T.W. Martin, A. Azemi, D. Hewett, and C.P. Schneider, "PSpice in Electrical Engineering Laboratories," *Proceedings of the 1992 ASEE Annual Conference*, pp. 1307-1308.

[2] D. Andrews, A. Azemi, S. Charlton, and E. Yaz, "Computer Simulation in Electrical Engineering Education," *Proceedings of the 1994 ASEE Gulf-Southwest Section Meeting*, pp. 77-82.

[3] A. Azemi and E. Yaz, "PSpice and MATLAB in Undergraduate and Graduate Electrical Engineering Courses," *Proceedings of the 24th Frontiers in Education Conference*, pp. 456-459, 1994.

[4] E. Yaz and A. Azemi, "Utilizing MATLAB in two Graduate Electrical Engineering Courses," *Proceedings of the 25th Frontiers in Education Conference*, pp. 2c6.1-2c6.4, 1995.

[5] A. Azemi and C. Stook, "Utilizing MATLAB in Undergraduate Electric Circuits Courses," *Proc. of the 26th Frontier in Education Conference,* Salt Lake, UT, vol. 1, pp. 599-603, 1996.

[6] A. Azemi and E. Yaz, "Utilizing SIMULINK and MATLAB in a Graduate Nonlinear Systems Analysis Course," *Proc. of the 26th Frontier in Education Conference*, Salt Lake, UT, vol. 1, pp. 595-599, 1996.

[7] A. Azemi, and E. Yaz, "Using MATLAB in a Graduate Electrical Engineering Optimal Control Course," *Proceedings of the 27th Frontier in Education Conference,* pp. 13-17, 1997.

[8] A. Azemi, and E. Yaz, "Using Graphical User Interface Capabilities of MATLAB in Advanced Electrical Engineering Courses," *Proceedings of the IEEE Control and Decision Conference*, pp. 4549-4554, 1996.

[9] R. Dorf, and R. Bishop. *Modern Control Systems*, ninth edition. Addison-Wesley Publishing Company, 2001.

[10] G. Franklin, J. Powell, and Emami-Naeini. *Feedback Control of Dynamic Systems.* fourth edition. Prentice Hall, 2002.

[11] M. Roden. *Analog and Digital Communication Systems.* Discovery Press, 2003.

[12] S. Haykin. *Communication Systems,* fourth edition. John Wiley, 2001.

[13] J.G. Proakis, and M. Salehi. *Contemporary Communication Systems Using MATLAB.* Brooks/Cole Publishing Company, 2000.

[14] L.H. Turcotte, and H.B. Wilson. *Computer Application in Mechanics of Materials Using MATLAB.* Prentice Hall, 1998.

[15] D.M. Auslander, J.R. Ridgely, and J.D. Ringgenberg. *Control Software for Mechanical Systems: Object-Oriented Design in a Real-Time World.* Prentice Hall, 2002.

[16] P.I. Kattan. *MATLAB Guide to Finite Elements.* Springer-Verlag, 2003.

[17] B.D. Harper. *Solving Statics Problems in MATLAB.* John Wiley, 2002.

[18] R.C. Gonzalez, E. Woods and S.Eddins. *Digital Image Processing Using MATLAB,* Prentice Hall, 2004.

[19] D.T. Kaplan. *Introduction to Scientific Computation and Processing.* Brooks/Cole, 2004.

[20] M.E. Herniter, and D.S. Scott, "Teaching Programming Skills with MATLAB," *Proceedings of the 2001 ASEE Annual Conference*, session 1520.

[21] P.E. Devnes, "MATLAB and Freshman Engineering," *Proceedings of the 1999 ASEE Annual Conference*, session 3353.

[22] W.J. Palm III. *Introduction to MATLAB 7 For Engineers.* McGraw-Hill, 2004.

[23] A. Gilat. *MATLAB: An Introduction with Application.* John Wiley, 2004.

[24] F. Gustafsson and N. Bergman. *MATLAB for Engineers Explained*. Springer-Verlag, 2003.

[25] David C. Kuncicky. *MATLAB Programming*. Prentice Hall; 2004.

Course Schedules

Table 1: First Delivery Course Format: *Fall 2004 Schedule (**partial table**; full table can be accessed at: http://www.engr.de.psu.edu/cmpsc201/fall04/cmp201_m.htm)*

Week	Activities	Assigned Problems
1	Course outline; History of computing; Inside a computer; Programming and Programming languages unit1.doc; Basics of the C++ environment; unit 2_1.doc; Working with the C++ compiler; cout, data type, arithmetic operators, formatting, variable declaration;	Handout
2	Interactive C programming and Handyboard. **C++:** Variable declaration; Quiz #1;	Handout
3	**C++:** Variable declaration (cont.); **unit2_2.doc** Interactive C programming and Handyboard. **C++:** Assignment & interactive input; unit3.doc Quiz #2;	Handout
4	**C++ functions;** Functions-1.doc; Interactive C programming and Handyboard. Example Programs. **C++ functions (cont.);** Functions-1.doc; **Quiz #3;**	Handout
5	**Chapter 1**: overview of Matlab: unit1.ppt Interactive C programming and Handyboard. **Quiz #4**	1.23; 1.27; 1.28 1.29; 1.30
6	**Chapter 2:** 2.1-2.2; unit2.ppt Interactive C programming and Handyboard. **Chapter 2:** 2.3-2.5 **Quiz #5;**	2.20; 2.21 2.25; 2.26 2.37

Table 2: Second Delivery Course Format: *Summer 2005 Schedule (**partial table**; full table can be accessed at: http://www.engr.de.psu.edu/cmpsc201/su05/cmp201_m.htm)*

Week	Day	Date	Activities	Assigned Problems
1	M	May 16	Course outline; History of computing; Inside a computer; Programming and Programming languages. **Chapter 1:** Overview of Matlab **Chapter 1 (cont.):** The programming elements of Matlab; program documentation. **Problem Solving**	1.23; 1.27; 1.28 1.29; 1.30
1	T	May 17	**Chapter 2:** 2.1-2.2 **Chapter 2:** 2.3-2.5 **Problem Solving** **Quiz 1**	2.20; 2. 21; 2.25; 2.26

1	W	May 18	Chapter 2: 2.3-2.7 **Chapter 3: Functions and Files** **Problem Solving** **Quiz 2**	2.37; 2.42; 2.46; 2.49
1	R	May 19	Chapter 3 cont.; Control Structures Functions and Control Structures **Problem Solving** **Quiz 3**	3.14; 3.17 (DUE 5/23)
1	F	May 20	Chapter 3 cont.; Control Structures Control Structures (cont.) **Problem Solving** **Quiz 4**	4.17; 4.18 (DUE: 5/23)

Table 3: Third Delivery Course Format: *Fall 2005 Schedule (**partial table**; full table can be accessed at: http://www.engr.de.psu.edu/cmpsc201/fall05/cmp201_m.htm)*

Week	Day	Date	Activities	Assigned Problems
1	T	Aug. 30	Course outline; Introduction to robotics, Interactive C, and Handyboard. Pass out the Lego part list inventory. Setting up the teams (due 9/6/05) Have the links ready for downloading E-educator and IC.	Lego kits inventory **Weekly HW: Handout** **DUE: 9/6/05**
1	W	Aug. 31	History of computing; Inside a computer (20 page) Programming and Programming languages: Unit1.doc (5 pages) Program Check List & program Elements	
1	F	Sept. 2	**Chapter 1: Overview of Matlab** ➢ **1.1:** Matlab Interactive Sessions ➢ **1.2:** Menus and the Toolbar ➢ **1.5:** Matlab Help System ➢ **1.4:** Controlling Input & Output ➢ **1.4:** User Input	
2	M	Sept. 5	**Labor Day Holiday**	
2	T	Sept. 6	Collecting the Lego inventory lists. Steps required for design, building, and testing of a robot Description of the course project and the "Rescue Mission" project IC programming unit 1 HW#1: Hello world type programming	**Weekly HW: 2.20; 2.22; 2.25; 2.26** **DUE: 9/12/05** **Lab HW#1** **Due:9/13/05**
2	W	Sept. 7	*Matlab data types* **Chapter 2: Numeric, Cell and Structure Arrays** ➢ **2.1:** Arrays (11 pages) ➢ **2.2:** Multidimensional Arrays (3 pages)	

			➢ *First Look at plots* **Quiz #1**	
2	F	Sept. 9	*Formatted outputs* **Chapter 2: Numeric, Cell and Structure Arrays (cont.)** ➢ **2.3:** Element-by element operations (14 pages) ➢ **2.4:** Matrix operation (10 pages) Helping with the homework	
3	M	Sept. 12	**Chapter 2: Numeric, Cell and Structure Arrays (cont.)** ➢ **2.5:** Polynomial Operation Using Arrays (5 pages) ➢ **2.6:** Cell Arrays (5 pages) **Quiz #2**	**Weekly HW: 2.37; 2.42; 2.49 DUE: 9/19/05** **Lab HW#2 Due:9/20/05**
3	T	Sept. 13	IC programming HW #1 due Different motors: DC, Stepper and Servo IC programming: motors	
3	W	Sept. 14	**Chapter 2: Numeric, Cell and Structure Arrays (cont.)** ➢ **2.7:** Structure arrays (6 pages) **Chapter 3: Functions and Files** ➢ **3.1:** Elementary Mathematical Functions (7 pages) ➢ **3.2:** User-Defined Functions: functions without arguments	
3	F	Sept. 16	**Chapter 3: Functions and Files (cont.)** ➢ **3.2:** User-Defined Functions: functions with arguments	

Sample Problems:

Sample #1: Linear Algebra

Write a Matlab program that will display a menu, which will let the user to:

1. Enter coefficients of a system of linear equations *[In this part you should first ask for the number of equations, number of variables and then their corresponding coefficients.]*
2. Find out if the system has solution and the nature of the solution (exact, infinite, or no solution), and display appropriate messages or give out the solution.
3. Exit

Make sure:
Follow the "Program Elements" guidelines*.

Sample #2: Integration of a user defined polynomial or function

Write a Matlab program that will display a menu, which will let the user to:

1. Enter coefficients of a polynomial [*This part should ask for the degree and then for the coefficients.]* or a function in string format.
2. Integrate the polynomial or the function; limits are entered by the user.
    a. Different integration schemes should be presented to the user as submenu for this part. If using Simpson, Lobatto, or Trapezoid the user enters the limits of integration, and error tolerance.
3. Integrate the polynomial or function.
4. Exit

Make sure:
Follow the "Program Elements" guidelines*.
Sample run for both polynomial and function should be included.

Sample #3: Differentiation of user defined polynomials or functions

Write a Matlab program that will display a menu, which will let the user to:

1. Enter coefficients of up to two polynomials or two functions in string format [*This part should first ask for the number of polynomials, then their degree and finally for the coefficients.]*
2. Differentiate each polynomial.
3. Plot the polynomials and its derivative on the same graph *[use subplots if two polynomials choice was selected]*
4. Obtain derivative of the quotient of the polynomials *[if two-polynomial choice is selected]*
5. Exit

Make sure:
Follow the "Program Elements" guidelines*.
Sample run for both polynomial and function should be included.

***"Program Elements"** guidelines described a typical Matlab program with expected documentations.