Learn MATLAB piggybacked onto C-programming

Dr. MADDUMAGE KARUNARATNE, University of Pittsburgh, Johnstown

Dr. Maddumage Karunaratne is an Associate Professor and the Head of the Electrical Engineering Technology department at the University of Pittsburgh at Johnstown, PA. The department offers undergraduate degrees in Electrical Engineering Technology and Computer Engineering Technology. Dr. Karunaratne earned a Bachelor of Science degree from the University of Moratuwa (Sri Lanka), a Master of Science from the University of Mississippi (Oxford), and a Ph.D. from the University of Arizona (Tucson).

Before joining academia, he gained fourteen years of extensive industry experience working in the semiconductor industry performing software development, application engineering, design, testing and verification of digital integrated circuits. He has taught electrical and general engineering technology classes at Pitt-Johnstown since 2004.

His research and teaching interests include Semiconductor circuit Testing and Verification, Low Power Design Analysis, Digital and Embedded Systems, and Electronic Design Automation. He is an author of over 23 publications and a US patent holder.

He can be reached at maddu@pitt.edu 225 Engineering and Science Building University of Pittsburgh at Johnstown Johnstown, PA 15904

Learn MATLAB piggybacked onto C-programming

Abstract:

It is apparent that the electronic industry is making capable and inexpensive consumer devices as evident from smart phones and tablets that are coming out to the market at an accelerated phase. For individual developer or capable consumer those devices offer customization to a level that was never seen in electronics. However, such customizations require development of computer programs that would control the devices. When electrical and/or computer engineers are trained, it is becoming more imperative that nearly all acquire solid programming skills to effectively function as electrical and/or computer engineers in their careers.

At the University of Pittsburgh Johnstown campus, electrical engineering technology (EET) and computer engineering technology (CET) majors always take one semester course on computer programming so they can be effective in using embedded controllers and other programmable devices, later in their curricula or in industry after graduation. It is a C language based programming course with few programming projects appropriate for the second year engineering students. However, several of their follow on courses require them to use MATLAB as a problem solving tool in advanced circuit theory and control systems courses. Students have been learning basic MATLAB on their own, and then learn advance features such as control and signal processing toolboxes with help from instructors in follow-on courses. Instructors in the upper level classes can only make limited efforts to help students learn MATLAB. Their efforts are geared toward actual subject contents which are heavy in abstract concepts and mathematics.

This paper discusses the experience in introducing MATLAB as an additional programming tool to sophomore level students who are learning programming in C language as their main objective. The author is introducing MATLAB in C programming course with the intent of reducing their future burden of learning its basics on their own. However, they learn advanced concepts and toolboxes in higher level courses. An additional benefit to this class is that MATLAB reinforces the concepts taught in C programming such as loops, indexing, conditionals, input/outputs, storage management, data and program structures, etc. Also, learning it at this stage to create visual effect based programs provides incentives and variety to further practice algorithm development and problem solving skills by students. Practice of such skills is essential to become competent programmers.

The paper also presents results from student surveys taken at the end of the course to gauge the student experience on learning an additional programming tool. The author intends to follow through the surveying based data collection in to the future semesters of the same student group when they use MATLAB for follow-on courses in their majors. Such findings will be published in an appropriate venue in a future time.

part I: introduction:

Since the tiny transistor was invented many decades ago, electronic industry continues to make vast strides in providing functionally rich inexpensive devices for consumers and industries. In recent years that trend has even accelerated due to very inexpensive integrated circuit manufacturing technology

which can now implement entire electronic systems in a single integrated package containing over billion transistors. Such inexpensive devices have been proliferated in to almost every consumer device, equipment, or instrument that can provide some level of electric power. Another trend has started very recently to utilize the telephone and other communication devices to control and monitor others wirelessly using the Internet and its extensions. All types of communication devices have become popular among consumers, particularly college-age young.

Electrical and computer engineers will not be successful without having a good skill set in understanding how the communication devices work in terms of exchanging and processing data. While most of them may not become software application developers, they would still need to understand the trends, adapt, and adopt the technologies to be successful in their careers. Teaching embedded systems has been around for a long time for the purpose of giving the skills to develop such control and communication systems in both software and hardware. However the value of acquiring such skills has even been increased now that almost every controllable device, from expensive automobiles to inexpensive household LED light bulbs, contains embedded devices for customization and convenience of consumer. Some of them may even provide interfacing capability to smartphones via custom developed small software modules known as Apps. Such customizations provide ample career opportunities to electrical and computer engineers confident in computer programming. This adds further reasons to make these students confident computer programmers.

As has been the case in the beginning, electronic embedded systems eventually are constrained by power, memory, processing capacity and ultimately the overall efficiency. The programming language and the platform of choice is mostly C or C++ due to its efficiency in memory allocation, run time and ability to directly manipulate data in hardware components. Therefore, teaching C/C++ to electrical and computer engineering students is considered essential in any curricula. Once the solution is finalized in an algorithm, its software program may be developed for a specific platform such as MS Windows, Androids, etc. Such developments can be done in most software packages (CGI, MATLAB, JAVA, MS Studio, MS NET, etc.), but embedded devices are to be programmed by efficient languages like C/C++. For hardware (electrical and computer) engineers, their main programming domain still remains as C/C++, which is the main reason C language is being taught and required by EET and CET programs at this campus. Therefore, replacing C/C++ in this programming course with MATLAB or any other software development package is neither practical nor advantageous to students. Further, it is easy to learn Verilog Hardware Description Language in a follow-up course of the degree programs, when they have taken a C language course. Verilog grammar somewhat resembles C language syntax although semantics are vastly different in certain constructs.

Students have been learning basic MATLAB on their own, and then learn advance features in control and signal processing toolboxes with help from instructors in follow-on courses. They use Control toolbox for Control Methods course; Symbolic toolbox for Circuits II course; and Simulink for both of those courses. Instructors in the upper level classes can only make small efforts to help students learn basics in MATLAB. Their efforts are geared toward actual subject contents which are heavy in abstract concepts and mathematics. However, those toolboxes and features are required to solve problems in these courses.

This paper discusses the experience in introducing MATLAB as an additional programming tool to sophomore level students who are learning programming in C language as their main objective. The author has introduced MATLAB in C programming course with the intent of reducing their future burdens of learning basics it on their own. An additional benefit to this class is that MATLAB reinforces the concepts taught in C programming such as loops, indexing, conditionals, inputs, storage management, data and program structures, etc. Learning MATLAB during a basic C programming course provides incentives and variety to further practice algorithm and program development since MATLAB can be used to create visual effects and animation of self-created objects.

Students were to acquire and use the MATLAB textbook[15] for this course as well as a reference book in follow-on courses. Instructors in those courses can introduce toolboxes and provide handouts for their usage without loss in course time to teach writing MATLAB scripts and functions. Having to learn another programming (scripting) language along with C cannot be too significant an extra effort when both are being taught, provided the course content is managed properly. In order to gauge the perceptions, difficulty and effort levels, a survey was done at the end of the semester. It should be noted that the regular grading assessments also included MATLAB specific problems to keep students motivated in their learning. An incentive was the occasional hands-on sessions to show, tryout and practice visualization graphics and simple animations provided by the instructor. In a project toward end of the course, students demonstrated their grasp of programming constructs and also creativity in visual graphics.

Part II of the paper describes the course objectives and content along with the description of selected topics in C and MATLAB that were introduced; Part III elaborates on specific project assignments which were done in C and MATLAB; Part IV ponders upon the results of a survey answered by students in the course at the end of the semester to determine the added value of MATLAB as a learning aid; Part V states concluding remarks with future plans.

part II: course content and objectives

The objective of the course and the assignments were to provide students with a skill set so they would be able to analyze the broader picture, identify the actual problem, develop a solution, design and implement a software program as that solution. Students were not expected to partition the solution in to multiple code files, but they are to use structured forms in dividing the solution into distinct and coherent functions with data passed among them as parameter values. This is straight forward in both C and MATLAB. However, MATLAB allows multiple result values to be passed back directly whereas one has to use data pointers in C to pass back multiple values.

Table 1 below lists the course topics and the relevant objectives while stating whether MATLAB was discussed with regard to that topic or not. Whenever MATLAB was contrasted, its usage in writing code was assessed (in most cases) in a homework, a quiz, or in an exam to a lesser extent than its C counterpart. Author kept the rigor of MATLAB tasks to a degree less than that of C tasks in order to ensure the major emphasis was on C as the formal requirement of the course. In any case, topics or the depth of topics were not diminished by the added learning of MATLAB.

The course was not partitioned into C and MATLAB, but they were intertwined at each programming topic if deemed necessary. The course mainly followed the topics in the C textbook[14] while MATLAB textbook[15] was used to compare and contrast relevant topics. Table 1 shows that every C topic was not countered with contrasting MATLAB features. The reasons might be there was no difference except a trivial name difference (printf), MATLAB feature may not be relevant for them (binary bit wise operations), or the feature was missing (e.g. data pointers). The topics are organized into a sequential order as shown in Table 1 to provide a progressive learning flow. Frequently students were reminded that their future objective would be to use C and MATLAB as tools in higher level courses, and learning them "now" would make it easy in multitudes of futures courses that they are required to take for their degrees.

	Topic	Course Objective
1	Computing machines	Understand the basic architecture and data flow in a desktop PC
2	Number Systems	Understand the binary, hexadecimal, octal, and decimal number systems.
3	Data Types	Understand several C language data types, usage, and conversions
4	Operations	Understand arithmetic operators, basic input-output operations, and C language intrinsic functions; contrast with Matlab.
5	Decisions and Control	Understand C language operators and their use in implementing structured program control structures; contrast with Matlab.
6	Explicit operators	Understand bitwise and logical operators; compare with Matlab
7	File Access	Understand fundamentals of file access; contrast with Matlab.
8	Modular Programming & Functions	Understand and master the concepts of C based modular programming, their structures & activation, recursion, variable and function scopes, data persistence, etc.
9	Arrays	Understand structure of single and multidimensional arrays, indexing, table lookup; contrast with Matlab.
10	Pointers	Understand concept of pointer variables, usage, function/data access
11	Structures	Introduction to data structures, typedef, exposure to simple examples (array of data records, singly linked lists & insert/remove, sorted lists)
12	Program Design and Implementation	Demonstrate above by the design and implementation of C language programs in a contemporary Integrated Development Environment (IDE) – compile, build, debug tasks; contrast with Matlab development environment

Table 1: Course Objectives in Computer Systems, Programming, and Applications in C

Students learn the basic architecture of a modern computer in terms of the components that they already know, such as processor, hard drives, USB, keyboard, monitor, printers, clock speeds, and power. They are also given a quick historical background to awe them in the progress of the electronic computing industry. Number systems, data types, ASCII codes, and direct input/output formats (keyboard reading and displaying on monitor) are taught since they form the basic foundation of early programs they would develop.

They learn that the major difference is the compiled and interpreted nature of C verses MATLAB. The contrast between C and MATLAB widens with "explicit" operators, array

handling, and functions. They feel more comfortable in C functions - perhaps due to the fact it was the major emphasis in the course, but some find it difficult to grasp the idea of returning values to a caller in C due to its abstract nature (in MATLAB a variable is assigned the values that are to be returned). Interestingly, some students prefer to use global variables to pass a value to a calling function than the C-mechanism of "return." In order to reinforce generic programming constructs, the use of short-cuts in MATLAB (in arrays, input, output, and array operators) were discouraged.

Pointer concept is very new to them, but most of them grasp the concept quickly after several examples and demonstrations. It is noted that MATLAB does not have a concept of a pointer which limits its efficiency in writing very large software applications for complex problems due to memory storage management issues. MATLAB has its own dynamic binding and garbage collection mechanism which make it a bit slower in run time and less efficient in memory use due to having to maintain information on allocated memory space. The major advantage of MATLAB over C is in visualization for the students in this course as evident from the animation project described in Part III. The major difficulty students seemed to have had was switching between the syntax of C and MATLAB, particularly in the area of *arrays* and *for* loops. The last item in Table 1 is the usage of the features in the respective software development environments DevC/C++ and MATLAB. Rather than making it a single topic, it was taught hands on throughout the semester as the need and opportunity arose.

part III: course projects

Homework typically reflects on the material covered in the lecture class. For any portion of assignments that involve coding, students use DevC/C++ and MATLAB on Microsoft Windows based desktop computers in general computer labs. They develop the algorithms in most cases and only when the problem is deemed difficult for their level, the author provided a skeleton of the algorithm or the solution. Students were provided with example code segments and hands on practice code for certain topics in order to drive the ideas home.

About six weeks into the course, after students have practiced writing basic loops, case statements, arrays, strings, formatting and input/output functions, they were assigned the Calculator project to implement a subset of the functions available in the basic MS Windows based desktop PC calculator. The objective as stated on the assignment sheet is "To learn and practice C-programming by developing the basic standard calculator functions that are available on a personal computer desktop. It is a software tool without a graphical user interface (GUI)." The actual calculators on Window based systems use mouse and keyboard functions inside a Win32 programming environment. However, the course is the very first programming course and learning Win32 system programming at this level is neither feasible nor necessary.

The instructions given to students were "Consider that you are the calculator reading one character at a time. For each character you read, decide whether it is a digit (data), or a command (+,-,/,etc.). If it is a digit, you need to collect all the digits that come next before any other type character (that other type character could be a command). If a character you read is not a digit after reading a set of digits (except in case of backspace), then you need to convert it to an integer (or a float)." They were required to try out the windows PC calculator to understand how it works. The first part of the project

was to create a flow chart to show what to do when different commands and digits are read in by the calculator. The expectation was that they would also practice algorithm development at detail level, one character at a time to build numbers and commands. The instructor checked the first part before they were allowed to proceed to implement the algorithm in devC/C++.

Students were allowed to form groups of two or work alone on the project. Most of them formed groups rather than work alone and that arrangement worked well. Only a couple of groups needed help from the instructor in completing the assignment. Their deliverable, a working program, was assessed by the instructor for grading using a variety of input values and operations. The C programming project was a very good learning experience as evident from the survey results in Question 2 in Table 2, for it involved most of the topics they studied up to that point in class.

Since DevC/C++ does not offer visualization and graphical manipulations, a MATLAB based project was also given to them. It was to animate the movement of objects on the screen using the plot function without use of special Toolboxes which would have required a much greater effort to learn than the time permitted. Basic MATLAB functions were sufficient to make student practice and feel that they can have fun while learning. Matlab is very user friendly for data visualization purpose, although other software packages are better suited for commercial computer animations. The projects are to be designed with the limited time available to learn basics of programming and command syntax.

This project was to draw (plot command) an air plane, balloon, helicopter, bird, or superman, or any flying object on a fixed size figure (graph) window. It was to be smooth looking with sufficient number of points and scaled if needed. Students were provided with an example animation of an analog clock and a walking man as a start due to limited time. The project was assigned only three weeks before the semester ended. The animation was achieved by repeated plot command applied on the same data set (which draws the figure), but shifted by (Xo,Yo) coordinates, where (Xo,Yo) represents the new location of the figure. This can be duplicated into any number of moving figures by having different figures and different (Xo, Yo) locations. In order for human eye to see the movement smoothly, the flicker rate needs to be more than 27 frames per second. The visible delays are easily achieved using *pause* command in Matlab.

In order for them to have a game-like feel, a projectile was added as a second moving figure which tried to intercept the other moving object. User was given the control of the projectile parameters such as velocity and the angle of launch. The interception was conveyed by indicating an explosion (a flicker of yet another figure for a few seconds) if the two moving objects came visually close enough as calculated by the coordinate distances. Students were to provide the MATLAB script for the instructor to run and grade their work while enjoying their creativity brought out by this course. The moving objects ranged from a simple kite, to a diving car with wheels spinning, steering turning, and the car bobbling up and down indicating that student must have had many nested loops in the script. More students opted to do it solo showing their creativity and confidence than in the previous (Calculator) project done earlier in the semester.

part IV: student feedback on visualization

Students were surveyed at the end of the course to gauge the effectiveness of having MATLAB as an additional programming envionment in their learning in the course. Another aspect is to see whether they would prefer MATLAB over C in the programming course in their degree programs, because they need to learn C anyhow to be competent in future work in electronics systems. While they would reap the benefit of having learned C language immediately in several courses at junior and senior level, knowing MATLAB in the current semester would help them immediately in the following semster when they need to use MATLAB Control Toolbox for the Circuits II class. Additional benefits of introducing MATLAB here might be that they may learn the hard fact of life long learning when they have to learn more MATLAB in two more semsters before graduation.

The survey was conducted on a class of 19 students (a typical electrical engineering class has less than 20 students at this institution) at the end of the ET 0031 programming course, and all of the students (18) present that day participated. The small engineering program at this campus has three four degree offerings. Typically only EET or CET students take the ET 0031 C programming course and the class size varies from 16 to 20. There is only one ET 0031 course section. Therefore, it was not possible to have a parallel control group to gather data for comparison. The prior year students had already used MATLAB in a spring semester course many months before the author contemplated this change. Therefore, attempting to gather meaningful survey data from that group to compare with this group would not have been fruitful.

The survey results are summarized qualitatively in Table 2. The responses were given based on the typical rubrics of 1 to 5 (1-strongly disagree, 2-disagree, 3-neutral, 4-agree, 5-strongly agree). Column 2 of Table 2 lists what each question was trying to assess from students' perception and knowledge. The 3rd column shows the average rubric score given by students for each question. Final column in Table 2 attempts to give a qualitative meaning to the data in column 3 so that author can make qualitative adjustments to the course content and projects in the future.

Almost all the questions posed to students received an affirmative answer of "yes" in the qualitative score indicating that the course content is adequate for them to learn programming while keeping the emphsis on learning C- language rather than MATLAB commands. Item 5 indicating "no" is a form of a validation that by placing emphasis on C (over MATLAB) students' preferences can be channeled in the beneficial direction. It is interesting to note that in the questions posted in Table 2, with exemption of Question 5, no student selected *stronly disagree* as a choice. For Question 5, also no choice was indicated for *stronly agree* category while only 2 have said they would just prefer MATLAB over C. That preference is countered by five students each stating *disagree* and *strongly disagree* choices.

	Essence of the question asked	Average	Qualitative
1	Students learned C programming well in this course	3.6	yes
2	Calculator project helped students learn C programming in this course	3.7	yes

3	The moving objects and animation project helped students learn MATLAB in this	3.4	yes
4	C - programming done was challenging (made me think) but it was fun	3.7	yes
5	Students preferred MATLAB programming over DevC/C++ programming	2.3	no
6	Students like to think in details, hence C and MATLAB programming are OK.	3.7	yes

Table 2: Student Survey Results

part V: conclusion

The decision to introduce MATLAB in C-programming course at the University of Pittsburgh at Johnstown was made for the Fall 2012 semester. It was done with ample caution and care so as not to overload students having to learn MATLAB in addition to the already established work load in C programming. Learning C cannot be scaled back due to its necessity and this being the only formal programming course in the EET program. As the student survey indicated, the two competing programming tools were balanced right and this experiment will encourage the continuation of this practice by the author. In the future, author plans to explore the possibility of using MATLAB to produce C-code for sample programs done by students, interaction between DevC/C++ and MATLAB in terms of MATLAB invoking C functions and Dev-C invoking MATLAB functions. Any such addition has to be considered carefully to maintain the learning effort at par with other courses at sophomore level.

In conclusion, the incorporation of a certain amount of MATLAB coding into an introductory C-programming course brings both immediate and near future benefits for both students and instructors. Students enjoy the visualization and rich graphical interfacing functions that MATLAB offers as revealed by the animation project described in here while strengthening their programming skills learned in C language. As a result, they will be more comfortable in learing other features of MATLAB such as Control Toolbox usage along with polynomial handling for their follow on courses. Instructors benefit by having students who have already been exposed to basic MATLAB and hence can easily be stepped up to the next level of required functionality in MATLAB without allocating class or lab time for introductions. In the long term, students will benefit by having gone through several follow on semesters learning more MATLAB on their own which would instill the practice of lifelong learning making them successful electrical and computer engineering graduates.

references:

[1] Qin, Qing-Hua, Wang, Hui, "Matlab and C Programming for Treffitz Finite Element Methods," Taylor & Francis Publishing, July, 2008.

[2] "The Right Tool for the Job," http://hammerprinciple.com/therighttool/items/matlab/c 2012

[3] Eddings, S. L., Orchard, M. T., "Using MATLAB and C in an image processing lab course," Image Processing, 1994. Proceedings, ICIP-94., IEEE International Conference .

[4] Frontoni, E., Mancini, A., Caponetti, F., Zingaretti, P., "A framework for simulations and tests of mobile robotics tasks", Control and Automation, 2006. MED '06, The 14th Mediterranean Conference on Control and Automation.

[5] Nelson, M.L., Rice, D., "Introduction To Algorithms And Problem Solving", Proceedings 2000 Frontiers in Education Conference, Oct. 2001, Kansas City, MO.

[6] Raymond, D.R., Welch, D.J., "Integrating Information Technology And Programming In A Freshmen Computer Science Course", Proceedings 2000 Frontiers in Education Conference, Oct. 2001, Kansas City, MO.

[7] Ludi, S. Collofello, J., "An analysis Of The Gap Between The Knowledge And Skills Learned In Academic Software Engineering Course Projects And Those Required In Real Projects", Proceedings 2001 Frontiers in Education Conference, Oct. 2001, Reno, NV.

[8] Budny, D., et. el., "Four steps to teaching C programming," Frontiers in Education Conference, 2002

[9] Ageenko, E., La Russa, G., "A Visualization Toolkit for Teaching, Learning and Experimentation in Image Processing", Frontiers in Education Conference 2005.

[10] Aleksi, I.; Kraus, D.; Hocenski, Z., "Multi-language programming environment for C++ implementation of SONAR signal processing by linking with MATLAB External Interface and FFTW

[11] Fincher, S., "What are We Doing When We Teach Programming?," Proceedings 1999 Frontiers in Education Conference, No. 1999, San Juan, PR,

[12] Jermann, W.,H., "The Freshman Programming Course: A New Direction", Proceedings 1996 ASEE Annual Conference, June 1996, Washington, DC.

[13] Westbrook, D.S., "A Multiparadigm Language Approach to Teaching Principles of Programming Languages", Proceedings 1999 Frontiers in Education Conference, Nov. 1999, San Juan, PR

[14] "Programming in C," 3rd edition by Stephen G. Kochan, ISBN-10: 0-672-32666-3, Sams Publishing 2004

[15] "Engineering Computation with Matlab," 3rd edition by David Smith, ISBN-10:0132568705, Pearson Education 2012.