Experience Report: CS1 in MATLAB for Non-Majors, with Media Computation and Peer Instruction

Cynthia Bailey Lee
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, USA
clbailey@ucsd.edu

ABSTRACT
As computer programming is increasingly considered an essential literacy skill for all students, MATLAB courses in particular can play a role in introducing non-major students to a tool commonly used in many of their fields. This paper reports on our department's experience introducing a CS1 in MATLAB for non-majors course. The course assumed no prior programming experience and no training in linear algebra. Without linear algebra and without the ability to do domain-specific tailoring, we turned to Media Computation to contextualize the skills and motivate students. Media Computation is an approach to programming instruction that focuses on manipulation of visual, audio, and video media. The course design also featured the Peer Instruction lecture format, in which lectures are punctuated by frequent questions that students answer individually and in small groups. To our knowledge, this represents the first time that Media Computation and Peer Instruction pedagogies have been comprehensively adapted to a MATLAB course. This work shares selected materials designed for this course, and reports outcomes of the two terms the course has been offered.

Categories and Subject Descriptors
K.3.2 [**Computer Science Education**].

General Terms
Design, Human Factors

Keywords
Media Computation, Peer Instruction, Clickers, Classroom response, Active learning, MATLAB, CS1.

1. INTRODUCTION
Our work on an introductory MATLAB course began in close collaboration with the Cognitive Science department, who wanted a required lower-division preparation for their upper-division coursework that uses MATLAB to perform domain-specific tasks, such as statistical analysis of experimental data and simulations of cognitive processes (e.g. neural nets). This course was to redress concerns about poor student preparation in their upper-division courses, and low student satisfaction with existing options for attaining introductory programming skills. The course is also part of our department's movement towards centralizing introductory programming instruction within the department—rather than having such courses scattered across departments—and assuming a larger service role on campus. This shift sets the stage for computer programming to be considered a fundamental literacy skill for all majors and a component of any liberal education, a vision articulated as early as 1961 by Alan Perlis. [9,10]

The design constraints and goals for the course were as follows:

- Assume no prior programming experience.
- Assume no training in linear algebra.
- Proactively address differences demographics, self-efficacy, and long-term goals that distinguish a non-majors audience from a majors audience.
- Build a solid foundation of basic programming skills (variables, conditionals, loops, functions, etc.), with special emphasis on unique MATLAB features for matrix handling.
- Do *not* include topics specific to cognitive science.

Without the ability to assume students have a foundation in linear algebra, an interesting question for a MATLAB course, what should they *do* with MATLAB, then? And with the request that the focus be on foundational programming constructs, and not cognitive science-specific content, another interesting question was, how can we contextualize and motivate the material being presented in this course? Media Computation (MediaComp) provided an attractive answer to both these questions. However, no textbook or other materials introducing MATLAB from a MediaComp perspective were available. This paper presents our efforts in creating our own.

To help foster an atmosphere of community learning and supportive collaboration, keep non-major students focused and engaged with the course material, and emphasize "big picture" conceptual understanding that can form a foundation for lifelong computer literacy, we adopted the Peer Instruction (PI) lecture format.

In this paper, we first define MediaComp and PI and review the existing literature on their benefits—benefits we hoped to obtain for our students. Then selected course materials are introduced, demonstrating how even very early beginning MATLAB concepts can be accessibly introduced using MediaComp and PI. Next follows a summary of evidence of the success of the course design, including student survey results. Finally, we discuss lessons learned and concluding advice for instructors interested in MATALB courses targeted to non-majors that are effective and well received by students.

SIGCSE'13, March 6-9, 2013, Denver, Colorado, USA.

2. RELATED WORK

Programming courses for non-majors present a particularly fraught pedagogical design challenge. Previous work notes issues with *self-efficacy* in introductory programming courses for non-majors. [19] Self-efficacy is the belief in one's own competence and ability to attain a goal. Many students entering this course have expressed low self-efficacy in relation to anything having to do with computers.

MediaComp is an approach to introductory programming designed with outreach and appeal to non-major students in mind. [9,10] Developed by Georgia Tech faculty Mark Guzdial and Barbara Ericson, MediaComp frames computer programming as a skill that enables students' expressive manipulation of digital media. This coupling of creativity and computing provides students with the context to see the utility of course topics. MediaComp textbooks and other materials are available for Java [7], Python [8], and Alice (Java) [4] programming languages. Comprehensive MediaComp materials were not yet available for MATLAB. We did take inspiration from *Introduction to Scientific Computation and Programming* [12], which also served as the students' textbook for the course. The text introduces MATLAB in a conventional (non-MediaComp) format, but includes extensive image, sound, and other media manipulation content in later chapters.

The MATLAB MediaComp materials described in this paper consist of both programming exercises (labs and homework) and lecture materials. All lectures in this course were conducted using the Peer Instruction (PI) pedagogy [13]. The PI methodology is as follows: (1) Before class, students prepare by reading the textbook. (2) In class, the instructor presents students with several multiple-choice problems. There may or may not be periods of traditional lecture between the questions. (3) For each question, students begin by individually answering and "voting" their response. (4) Then students discuss the question in small groups, ideally coming to a consensus, and vote again. (5) The instructor leads a discussion with the whole class.

PI was created by physics professor Eric Mazur to address concerns he had about the failure of introductory physics courses to change the way students thought about the physical world in their daily lives. Non-majors students in particular could pass their exams by rote regurgitation of formulas, but failed to correctly reason about high-level conceptual questions.[2, 13] Effecting a lifelong change in the way students think about computing, the Mazur wanted to effect a lifelong change in his students' understanding of the physical world, is an important goal of this non-majors course. The crux of productive PI is well-designed questions, often called ConcepTests [13]. ConcepTests are unlike most multiple-choice exam questions in that the purpose is to teach and spark discussion, and to focus on core concept understanding rather than performance of formulaic procedures. After enjoying years of study and success in physics, studies of PI in computing are now being reported [1,3,14,15,16,17,20]. To our knowledge, there are no reports of PI use in a MATLAB course.

3. SETTING

This section outlines the conditions of the classes reported on in this paper.

3.1 Classes and Instructor

The instructor who designed the course has now taught it twice at a large research intensive public university in the United States. The winter class had an enrollment of 100, and the spring class had an enrollment of 131.

An i>Clicker brand clicker device was required. Like a textbook, students may purchase clickers at the bookstore for $20-40, and a majority had already obtained one for previous course(s) in other department(s). Clickers offer the instructor detailed record-keeping of student votes, and offer students vote anonymity to their peers; both are advantages over lower-cost alternatives such as colored index cards.

The class format was three 50-minute lectures and one 2-hour lab per week. Additional drop-in assistance from department tutors was available throughout the week in the computer lab.

3.2 Assessments

Summative assessment for the course consisted of three in-class written exams: two midterms and a final. In a beginning, non-majors course, adequate formative assessment and feedback is especially important for supporting student learning and calibrating student expectations. Four categories of formative assessment were used in this course: reading quizzes, PI questions in lecture, lab assignments, and homework assignments.

3.2.1 Reading Quizzes

Advance preparation for lecture is essential for students in PI classes, because they will be responsible to their groupmates for productive participation. Reading quizzes were 5% of the course grade and simply provided some enforcement of advance reading.

3.2.2 PI Questions in Lecture

Clicker participation comprised 5% of the course grade, and points were awarded for participation, not correctness. PI questions were the most timely source of formative feedback—students were able to measure their understanding of concepts just moments after they were taught. Students could also situate their performance in the range of scores in the class, because of the instantly tabulated and displayed graph of all students' responses.

3.2.3 Lab Assignments

Weekly lab assignments were guided explorations of new topics necessary for that week's homework. Pair programming was used for all lab assignments—sometimes randomly assigned and sometimes student self-selected.

3.2.4 Homework Assignments

Weekly homework assignments challenged students to complete a more independent and complex programming task, applying skills introduced in the lectures and lab. Pair programming was used for all homework assignments, using the same pairing as that week's lab. Homework culminated in a two-week art or animation project of each pair of students' own design.

4. COURSE MATERIALS

In this section, we share the syllabus design process, as well as MediaComp and PI content artifacts created for this course. Examples were selected to highlight materials about programming constructs that are characteristic to MATLAB (e.g. matrix indexing), as opposed to those in common with other languages already taught using PI and/or MediaComp (e.g. while loops). Source code solutions are provided for selected programming

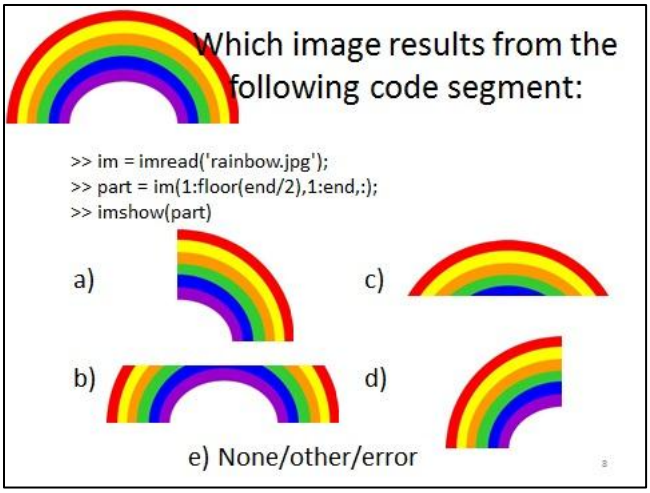

Figure 1. PI question on matrix row/column indexing.

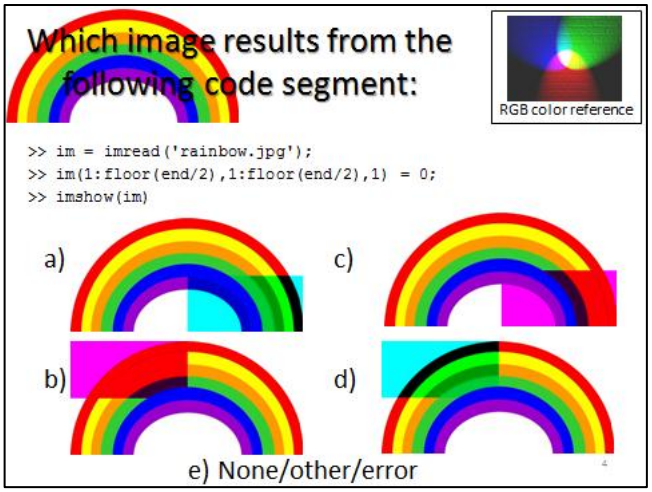

Figure 2. PI question on scalar assignment to a matrix region. ***(This figure best viewed in color)***

assignments to allow the reader to gauge the difficulty for students.

Additional PI materials for the course are available to instructors for non-commercial use at http://PeerInstruction4CS.org, under a Creative Commons license.

4.1 Syllabus Design Process

To apply MediaComp to a new language, MATLAB, the first task was to find a MATLAB textbook that teaches the content we wanted to teach, and a MediaComp textbook for a different language (Java) to provide project ideas, and try to reconcile the two. Starting with the Kaplan MATLAB text [12], we looked at the chapter (near to the end) where digital media is introduced. Taking the most basic exercises found there, we listed every skill necessary to minimally completing each of the exercises (e.g. call a function with an argument—the image file name, create a variable to save the output of a function, etc). Then we mapped out dependencies between all the skills on those lists and the skills leading up to those skills, and determined the shortest possible path to them. In other words, a single-minded emphasis on reaching key MediaComp-relevant skills as soon as possible dictated the ordering of topics at the beginning of the course. The result is to teach as much of the course as possible using MediaComp.

We determined that simple imperative programming turtle graphics commands could be introduced in the first lab (midweek in the first week of class), including allowing students to specify RGB hues to change the color of the turtle's line. Full RGB images (matrices) could be introduced as soon as the first lecture of the second week. Once RGB images were introduced, nearly all topics thereafter could be taught using MediaComp methods with RGB images. Green screen effects were introduced in the fourth week, and animation with green screen effects in the eighth week.

4.2 MediaComp Matrix Indexing

Although it has grown to encompass many things, MATLAB, as the name implies, specializes in handling of matrices of numeric data. Fluency with these features was a key learning goal for the course, enabling students to rapidly analyze data throughout their science careers.

4.2.1 PI lectures with MediaComp

MATLAB has a rich syntactic toolkit for matrix indexing by row and column, but it can take time for beginners to acclimate to mentally parsing everything that is going on in a line of code that does multidimensional indexing. Figure 1 shows a lecture slide that demonstrates how the synergy between PI and MediaComp creates a simple and effective way for students to work on this skill.

Throughout the course, we worked with RGB images, which, in MATLAB, are represented as three-dimensional matrices of type unsigned 8-bit integer ("uint8"). The first dimension is the rows, the second is the columns, and the third is the layers (always of size three—one layer for each of the red, green, and blue values in the RGB color scheme). MATLAB automatically decompresses JPEG and other image formats into full matrices of this format using the *imread* function, as shown in the code in Figure 1.

Matrix indexing syntax consists of comma-separated row, column, and layer subscripts. (In MATLAB, these are encased in parentheses, not square brackets, as array indices are in languages such as C/C++.) The colon operator is used to specify a range of values, and the *end* keyword means the last cell in that dimension. So the code snippet in Figure 1 selects the top half of the rows, all the columns, and all three color layers (colon operator by itself is equivalent to *1:end*), giving an answer of (c).

Figure 2 shows a similar question, this time with the matrix indexing occurring on the left hand side of the assignment operator. Students needed to puzzle out both which row/column region of the matrix was being selected, and what should happen to the color in the image when that region is set to zero. The selected region is the first (red) layer of the top left quadrant of the image. The *RGB color reference* on the slide (a recurring image) helped students recall that saturated green and blue together, with no red, gives teal. So the correct answer is (d).

These questions show how MediaComp supports implementation of the PI ConcepTest philosophy by enabling instructors to write questions that are stripped of distracting details, and engage students directly with important "big picture" concepts in the course. With a firm grasp of the big picture in hand, students can go forward to master the details—still a critical aspect of computer programming—in lab and homework settings.

Figure 3. *MakeWarhol* homework assignment output. *(This figure best viewed in color)*

4.2.2 Assignments with MediaComp

The matrix handling theme was carried over from lecture to the lab and homework assignments, where students wrote their own code. Figure 3 shows a student's project from the 3rd week of the term. (Student work shared with permission.) The image was created using what we call the *MakeWarhol* function: it takes as input an RGB image matrix, extracts the individual RGB layers, and tiles them (plus the original) onto the four quadrants of a new RGB image. The result is something loosely inspired by Andy Warhol's colorful repeating tiled images. Even more Warholesque results can be achieved by selecting pairs of layers (e.g. red-green) and applying a negative effect to some or all of the quadrants, things some students experimented with.

After writing the required code for assignments like this, students ran the code on images of their choosing, and posted the results on the course discussion board. This provided a chance for students to express their identity to other students, and to build a sense of accomplishment and community.

The solution code for MakeWarhol is as follows:

```
function res = MakeWarhol(im)
  red = im(:,:,1);
  green = im(:,:,2);
  blue = im(:,:,3);
  res = zeros(2*size(im,1),2*size(im,2),3);
  res(1:end/2,1:end/2,2) = green;
  res(1:end/2,end/2+1:end,1) = red;
  res(end/2+1:end,1:end/2,3) = blue;
  res(end/2+1:end,end/2+1:end,:) = im;
end
```

Other matrix-indexing homework and lab exercises included using the increment specification option for the colon operator to select every other row (or column), or every third row (or column), to achieve proportional or out-of-proportion image resizing effects. This code shows an image that is one third as tall as the original:

```
>> im = imread('rainbow.jpg');
>> imshow(im(1:3:end,:,:);
```

In the 2nd week assignment, students wrote code that performed a similar static-ratio resizing of an image. In the 3rd week assignment, students generalized their code by packaging it as a function, and parameterizing the horizontal and vertical resize factors.

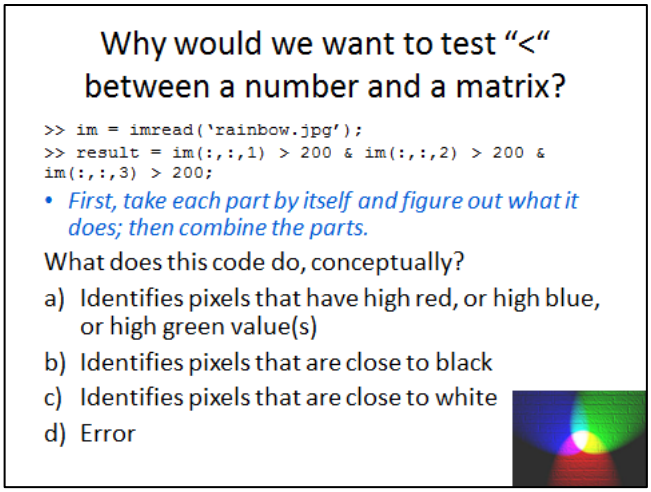

Figure 4. Selecting image regions by hue using logical matrix indexing.

4.3 MediaComp Logical Indexing

One of the most powerful features of matrix handling in MATLAB—and one of the most challenging for students to understand—is *logical indexing*. This feature allows users to index a vector or matrix of data using a matrix of type logical (i.e. true/false), rather than the row/column ranges shown in the examples above. A cell of the matrix is selected iff the corresponding cell in the logical matrix is true/1, and is not selected if the corresponding cell in the logical matrix is false/0. This feature allows users to easily create filter views of data—for example, from a matrix of health data, select the systolic blood pressure for only those individuals whose age is between 50 and 65.

It also happens that the feature is well suited to doing *green screen* special effects. Green screen effects (also called *chroma key* effects) are used widely in digital video production, from TV news to advertising to Hollywood films. A scene is filmed or photographed with elements (often a background) that is a very specific hue (often bright green). Later, during digital post-production work, that region of the scene will be selected by hue and replaced with some other image.

4.3.1 Creating matrices of type logical in PI

In lecture on the day prior to the lab on green screen effects, the use of logical operators (and/&, or/|) and relational operators (<,>,<=,>=,==) on matrices is introduced. MATLAB allows relational operators to be used between a matrix variable and a scalar variable. The output is a matrix where each entry is the result of the pairwise comparison between the scalar and the corresponding cell in the input matrix. This is illustrated in the PI question shown in Figure 4. (The answer is (c)—the closer the three component colors are to 255, the closer to white the color is.)

The purpose of this question was to introduce the syntax and its functioning, but also to start a dialog about why a user would want to do such a thing. The green screen concept was introduced with examples from popular media. Students were then prepared to create their own hue-selecting filters in lab.

4.3.2 Logical indexing lab assignment

In lab, students were provided with three images of equal size: a man with a red and black shirt on a white background, a telescope photograph of a starry sky, and a multicolor "tie-dye" pattern. Students were to select the man—but not his white background—and overwrite it with the space photo, creating a new background. For this task, they could use the selection code from class (Figure 4), with slight adaptations. The code from class used a very forgiving definition of "close to white." Using this included not only the pure white background, but also other white objects such as the man's teeth. Students can adjust the threshold to include a very small margin, or even only include pure white.

Next, they were to change the man's red shirt fabric to tie-dye by selecting the red part of the shirt and overwriting it with pixels from the tie-dye image. This required more careful hue thresholding calibration. The shirt has wrinkles and shadows, and even the best solutions include some false positives and false negatives in identifying all the red shirt pixels.

That students were largely content spending lengthy amounts of time writing and re-writing compound logical expressions and conditional statements illustrates the power of MediaComp as a motivating context for student exploration of computing. In course evaluations, many students mentioned the green screen lab as a compelling one, and in MATLAB the code was straightforward enough to use as soon as week 4. The solution code is as follows:

```
>> person = imread('person.jpg');
>> sky = imread('sky.jpg');
>> dye = imread('dye.jpg');
>> filter = person(:,:,1) == 255 &
person(:,:,2) == 255 & person(:,:,3) == 255;
>> filter = cat(3,filter,filter,filter);
>> person(filter) = sky(filter);
>> filter = person(:,:,1) > 120 &
person(:,:,2) < 60 & person(:,:,3) < 60;
>> filter = cat(3,filter,filter,filter);
>> person(filter) = dye(filter);
```

For most assignments, students were encouraged to provide their own images as input to their code. This allowed students to spend time with images that are meaningful to them. For this assignment, we provided the three specific input images because the code only works for images that have exactly the same dimensions. Adjusting for different sizes adds several steps, something students were able to master soon after mastering this lab assignment. Students made pervasive use of the different-size image green screen effects on their final projects in the course.

5. RESULTS

University evaluations show the average student approval of the course that existed before this one was 73% (14 classes offered since 2007). For the new course, student approval was 81% the first time it was offered, rising to 88% the second time. There was a low rates of students dropping the class, relative to the average for the department, 5% of of the first-day enrollment in Winter and 8% in Spring. Comments from students on the evaluations were overwhelmingly positive, many validating achievement of key design goals for the course (these comments summarized below). By all anecdotal accounts, both the computer science and cognitive science departments have been pleased with these outcomes.

There are three sources of information for evaluating the students' perspective of whether the course design was successful in the two times the course has been offered thus far (Winter 2012 and Spring 2012 terms). First, we have the usual university-conducted end of term course and professor evaluations quoted above (64% and 38% response rates, respectively[1]). Second, we conducted additional attitudinal surveys of the students directly soliciting their feedback on use of PI (94% and 83% response rates). Third, we have an attitudinal survey about lab and homework design (Winter only, 75% response rate). Student quotes from these three sources are combined and presented below by topic.

5.1 Lab and Homework Assignments

In the homework survey, students reported an average of 5.6 hours per week of outside study for the course (stdev=2.7, max=12.5 hours), which is within the department's guidelines for appropriate student workload. When asked to name the "most rewarding" assignment, "whether it was easy or hard," the most commonly listed assignments were a tile puzzle game[2], MakeWarhol, and the green screen assignment (collectively accounting for about 70% of responses).

For the MakeWarhol assignment, even a few students who mentioned struggling with it said it was still the most rewarding in the end. One student said, "I use the picture we made with the MakeWarhol code as a background on my computer." Another student said, "MakeWarhol because we did something with an image that we might actually want to do sometime for a project or just for fun."

5.2 MediaComp Pedagogy

Though not everyone agreed ("I think this…course should teach more about how to apply it to math and data rather than images"), most students reported extremely positive reactions to the MediaComp approach used throughout the course. "I have never taken a [computer science] class before and I was really worried because I had heard about how difficult [computer science] was but you have not only made it easy to understand but fun as well."

Students valued the way the MediaComp approach provided immediate visual feedback during debugging. One student noted, "I think the ones that involve images are most rewarding because the results are very apparent when they are done correctly." Another student said, "It was nice to be able to implement code to [do] something more interesting, [it] helped make it easier to understand how the code worked."

Some did see a connection between skills developed using MediaComp and more traditional applications of those skills. A student commented, "The course was interesting. Although most of the programming was done with media the processes that we learned are very applicable to data analysis."

Students also enjoyed how working with digital media gave them a new, insider's view of computing in their daily lives. One student said, "Interesting course, I enjoyed that it dealt a lot with manipulating images with code—showing us the inner workings of functions we use at the click of a button with programs like

[1] The median response rate at the university is 28%.

[2] The tile puzzle game took as input any RGB image, divided the image into a 4x4 grid of tiles, mixed up the order of the tiles (removing one to create an empty space), and let the user play an interactive game of "sliding" the tiles back into the correct order.

photoshop, etc." Another student declared, "I have new found respect for computer programers [*sic*]."

5.3 PI Pedagogy

On the PI survey, 87% and 83% of students (Winter and Spring, respectively) said they would recommend that other instructors adopt the PI method. One student expressed his or her perception of its impact on learning, "Looking back, it felt that I learned the material automatically by coming to class since it was an engaging class, instead of having to spend time after class and before exams reviewing the material as in normal lectures. Compared to other lectures, I felt like an active participant instead of a passive listener, and I much preferred this role as a student instead of the normal role since in the normal role I usually feel the need to fall asleep."

93% and 91% of students on the same survey agreed with the statement, "The immediate feedback from clickers helped me focus on weaknesses in my understanding of the course material."

6. CONCLUSIONS

We found MediaComp to be an effective pedagogical means to contextualize computing concepts and motivate non-major students in a MATLAB CS1 course. In this way, MediaComp has mitigated the problem created by our course design constraint that we not include cognitive science domain-specific applications of MATLAB in this course.

We have further found that using MediaComp contexts for Peer Instruction (PI) ConcepTest questions often creates a synergy between the two pedagogies. This is because ideal ConcepTests confront students with scenarios they can reason about and discuss at a high level, without unnecessary detail and process interfering. When the inputs to and outputs from a piece of code are concrete images, students have a powerful language of programmatic cause and effect to use in their peer discussions.

The success of these pedagogies in a MATLAB course is especially significant because of the role the MATLAB language plays as a commonly used tool for students, professionals, and academics in disciplines outside of computer science. If computer programming is to become a critical component of a liberal education, regardless of major field of study, languages such as MATLAB may become an important vehicle for that education.

7. REFERENCES

[1] Carter, P. An experiment with online instruction and active learning in an introductory computing course for engineers: JiTT meets CS. 14th Western Canadian Conference on Computing Education, 2009.

[2] Crouch, C. H., and Mazur, E. Peer instruction: Ten years of experience and results. Am. Journal of Physics 69, 2001.

[3] Cutts, Q., Carbone, A., and van Haaster, K. Using an Electronic Voting System to Promote Active Reflection on Coursework Feedback. In Proceedings of International Conference on Computers in Education, 2004.

[4] Dann, W. P., Cooper, S. P. and Ericson, B. Exploring Wonderland: Java Programming Using Alice and Media Computation. Prentice Hall. 2009.

[5] Greenberger, M. Computers and the World of the Future. Transcribed recordings of lectures held at the Sloan School of Business Administration, April, 1961. MIT Press, Cambridge, MA, 1962.

[6] Guzdial, M. A media computation course for non-majors. In Proceedings of the 8th annual conference on Innovation and technology in computer science education (ITiCSE '03), 2003.

[7] Guzdial, M. and Ericson, B. Introduction to Computing and Programming in Java: A Multimedia Approach. Prentice Hall. 2006.

[8] Guzdial, M. and Ericson, B. Introduction to Computing and Programming in Python: A Multimedia Approach. Prentice Hall. 2009.

[9] Guzdial, M. and Forte, A. Design process for a non-majors computing course. SIGCSE Bull. 37, 1. 2005.

[10] Guzdial, M. and Soloway, E. Computer science is more important than calculus: the challenge of living up to our potential. SIGCSE Bull. 35, 2. June 2003.

[11] Hake, R. R. Interactive-engagement vs. traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. Am. J. of Physics 66, 1998.

[12] Kaplan, D. T. Introduction to Scientific Computation and Programming. Cengage Learning. 2003.

[13] Mazur, E. Peer Instruction: A User's Manual. Benjamin Cummings. 1996.

[14] Pargas, R. P., and Shah, D. M. Things are clicking in computer science courses. In Proceedings of the 37th SIGCSE, 2006.

[15] Porter, L., Lee, C. B., Simon, B., Cutts, Q., and Zingaro, D. Experience Report: A Multi-classroom Report on the Value of Peer Instruction. In proceedings of the 16th ITICSE, 2011.

[16] Porter, L., Lee, C. B., Simon, B., and Zingaro, D. Peer Instruction: Do Students Really Learn from Peer Discussion in Computing? In proceedings of 7th International Computing Education Research Workshop, 2011.

[17] Simon, B., Kohanfars, M., Lee, J, Tamayo, K., and Cutts, Q. Experience report: Peer instruction in introductory computing. In Proceedings of the 41st SIGCSE, 2010.

[18] Smith, M., Wood, W., Adams, W., Wieman, C., Knight, J., Guild, N., and Su, T. Why Peer Discussion Improves Student Performance on In-Class Concept Questions. Science, 2009.

[19] Wiedenbeck, S. Factors affecting the success of non-majors in learning to program. In Proceedings of the International Workshop on Computing Education Research. 2005.

[20] Zingaro, D. Experience report: Peer instruction in remedial computer science. In Proceedings of the 22nd World Conference on Educational Multimedia, Hypermedia & Telecommunications, 2010.