Re-examining the Core Computation Skill Set of Civil Engineering Undergraduates

Abstract

The contents of the civil engineering curriculum related to computation are usually derived from a combination of ABET program requirements, topics found on the Fundamental of Engineering (FE) Exam, and topics from "legacy" courses that introduce engineering graphics and programming. These components have generally fulfilled the needs of CE majors by engendering basic skills with numerical analysis and graphical representation (i.e., CAD). However, as industry moves towards integrated virtual design and analysis methods, the need to reassess the computational curriculum to address this change is apparent.

In this paper we re-examine the historical basis for the core computational skill set and assess the degree to which it has met the needs of industry for modeling, design, and construction. We identify technology trends such as scripting languages (e.g., Python and Ruby), modern numerical programming (e.g., Matlab), and Building Information Modeling (BIM) tools (e.g., Revit and Sketchup) that, to be available and effective for industry, must have a curricular basis for CE graduates. Finally, we provide recommendations for incorporating modern tools into both introductory engineering computing courses and senior level design courses.

Introduction

Computational skills have been an integral component of engineering education for decades. Long before digital computer technology was available, computing by means of mathematical tables, physical devices (e.g., slide rules), and graphical tools (e.g., nomographs) was common and efficiently employed. Before the 1960s, computing tools served only one purpose: to automate mathematical calculations. The early age of the digital computer bifurcated this need for computing into distinct sub-areas including, but not limited to, numerical computing, graphical modeling, and data processing. Later ages have introduced network computing, synchronous and asynchronous communication, database modeling, and immersive virtual design environments. The extent to which engineering curricula have kept up with the constant changes in computing has been a frequent theme in the engineering education literature[1]. The purpose of this paper is to provide an up-to-date assessment, from the perspective on one civil engineering program, on the current computational skills and knowledge needed today, our experience with developing courses to deliver these skills, and our recommendations for future curriculum development to define and keep current the core computational skill set.

Previous Research

The undergraduate curriculum used by most U.S. civil engineering programs is closely aligned with the program criteria and outcomes defined by ABET[2]. The current program criteria define the minimum curriculum and faculty qualifications expected of accredited programs. The program outcomes define the knowledge and skills expected of graduates upon completion of the program. It is worth noting that within the current and, for that matter, recent program criteria,

computing is not explicitly mentioned as a required part of the curriculum. By contrast, the current program outcomes do allude to computing. Specifically, of the current program outcomes, outcome (k) "an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice" is frequently related to computing aspects of the civil engineering curriculum. This outcome has had a unique impact on the curriculum development related to computing. Instead of curricular requirements defined in absolute terms such as "conduct civil engineering experiments and analyze and interpret the resulting data," the program outcome (k) suggests that computing skills and knowledge be indexed to the needs of industry. Defining the current computing skills and knowledge needed in modern civil engineering practice has been the goal of numerous studies, which are described below.

Abudayyeh et al.[1] presented results of surveys conducted by the ASCE Technical Council of Computing and Information Technology Committee to assess the current computing component of civil engineering curricula. Data from the 2002 survey as well as past surveys from 1995, 1989, and 1986 were discussed. The surveys requested practitioners and educators to comment on the relative importance of specific computing applications and skills. Table 1 summarizes some of the surveyed computing topics.

Table 1: Summary of 2002 computing survey questions after Abudayyeh et al.[1]

Computer application	Computer-related skill
Project management	Use of spreadsheets
Simulation	Use of equation solvers
CAD/GIS	Programming in Fortran, C++, Java
Optimization	Use of electronic communication
Collaboration Environments	Use of database management systems

Table 1 is especially useful because it explicitly defines what the computing skills of value are (or were) at the time. Comparisons with the older surveys are somewhat limited due to changes in the survey questions and format, but all the surveys requested ranking of importance for spreadsheets, word processing, CAD, and programming. The surveys show that programming competence is ranked very low by engineering practitioners and the use of equation solvers and databases have declined over the past decade. The authors argued that in order to prepare graduates to operate effectively in the emerging and changing computing environment, adequate computer resources, expertise in teaching of computing, and an appropriate computing component are needed.

Garrett et al.[3] presented summary of ASCE 4th International Joint Symposium on Information Technology in Civil Engineering to develop a vision for the future use and development of information technology in the field. They discussed proposals for new directions in order to prepare the next generation workforce, which will be expected to multitask and be conversant with many types of information technology. The paper highlighted that most CE students are incorrectly convinced that they need not concentrate on computing technologies because it will be unlikely that they will ever have to write code. One of the recommendations of the paper is that CE students need a theoretical undergraduate course in computing using basic engineering examples and exercises as a baseline, even if they never apply those skills later in their career.

Some other previous works investigate the issues of computing pedagogy in civil engineering education. Grigg et al.[4] points out that there is no single best way to teach computer-based tools for civil engineering students. They argue that a specific tool, such as AutoCad™ or Matlab™, can be taught intensively in a single course, but unless subsequent instructors require its use, student proficiency in the software will be lost. If a software tool is covered progressively in courses, each course instructor must know the skill sets the students have acquired. Abudayyeh et al.[1] make this point as well.

El-Zein et al.[4] discussed the move by engineering schools towards a first-year computational class. Their paper discusses the design, development, and implementation of an e-learning tool into a freshman course and offers a student-centered model for integrating e-learing with face-to-face interaction. They devised a course made up of two components: CAD with SolidWorks™ and programming using Matlab. The authors claim that the course provided students with more learning resources.

These studies, as well as others, highlight two important points. First, the relative importance of computing knowledge and skills is not static. As new technologies emerge, accreditation authorities and the practicing community expect that these technologies will be incorporated into the curriculum. Second, the success with which computing topics are covered depends in part on the degree to which they are reinforced throughout the curriculum. Casey[6] expands on the issue of reinforcement in the incorporation of Building Information Modeling (BIM) in the civil engineering curriculum both in introductory courses and in modules used in subsequent courses.

Perspectives on Computing for the Millennial Generation

Those individuals born between the early 1980s and the mid-1990s define the Millennial Generation. They represent the current cohort of civil engineering undergraduates. Their attitudes toward computing are quite different from the students that preceded them. For example, millennial generation students have never been without the Internet, and in urban and affluent areas, never without broadband internet. They identify intrinsically with social networking applications such as Facebook and Twitter. They have never had to type commands at a prompt or learn any kind of language in order to use a computer. To them, computers have always been visual tools. Computer games and animation have instilled in them a comfort with virtual representations of the physical world that prior generations lack.

These attitudes explain to a certain extent the changing attitudes about computing in civil engineering education. Millennial students have predominantly been users of software, not creators of it. They are attracted to applications that combine visual representation with analysis, GIS and BIM being good examples of this. They approach design in a holistic manner, considering visualization and inter-dependency up-front, rather than at the end of the process. These changing attitudes, in addition to the motivation brought by the practicing community, dictate a re-examination of the core computational skill set reflected in the civil engineering curriculum. The sections that follow dictate our experience in updating our computing and IT curricula to reflect the expectations for accreditation, professional practice, and student attitudes.

Developing a modern Engineering IT course

Since the inception of the BS in Civil and Infrastructure Engineering program at George Mason University, the need for an introductory computing experience for freshmen has been met by the introductory programming course offered by our Computer Science Department, CS112 – Introduction to Computer Programming. CS112 is a first course in programming designed for CS students and has taught, since the mid-1990s, predominantly object-oriented programming techniques with C++, Java, and most recently Python. The course is required for all CE students, and from the beginning has caused difficulties.

First, the pass rate for CE majors in this course has been noticeably low. Since 1997, the proportion of students failing CS112 who were CE majors was greater than any other [declared] major. Second, the performance of CE students on the computing portion of the Fundamentals of Engineering (FE) examination has been particularly low. The FE computing questions, which comprise up to 7% of the morning (general) session include topics such as: terminology (e.g., memory types, CPU, Internet); spreadsheets (e.g., addresses, interpretation, "what if," copying formulas); and, structured programming (e.g., assignment statements, loops and branches, function calls).

These topics, which might seem elementary, have been absent from the curriculum for most instances of the CS112 course. Instead, the course has favored the more nuanced aspects of object-oriented design. These deficiencies, in addition to the perceived need for adapting the computing curriculum for civil engineering, have motivated the development of a new IT for Engineering course beginning in 2007. The new course, ENGR117, was offered for the first time in spring 2009, was designed to meet these objectives:

- understanding the historical context for computing in scientific and engineering applications;
- fluency in spreadsheet analysis including the use of formulas, charts, data analysis, and solvers;
- competency in computing and programming fundamentals such as data types, variables, program flow, input/output, and procedural methods;
- comprehension of the advantages and disadvantages of scripting over compiled languages and their use for automating tasks in science and engineering;
- creation of a foundation onto which computing knowledge and skills may be applied to future coursework, applied on engineering licensure examinations, and used in professional practice;
- develop an aptitude in 3D modeling and apply scripting to enable behavior and properties to 3D models of engineering objects;
- committing to an attitude of life-long-learning with technology.

The last objective is noteworthy because it coincides exactly with ABET outcome (i). Each objective was intended to address the deficiencies previously discussed and to adapt to modern trends in the engineering and construction industry. The course has been implemented in three parts, with modules in subsequent engineering courses as described in the following sections.

Advanced spreadsheet models

Spreadsheets are perhaps the most widely used computational tool amongst civil engineering curricula. And according to the surveys conducted by Abudayyeh et al.[1], still the most valued by practitioners. The capabilities of spreadsheets have advanced considerably in the past decade to include: equation solvers and optimization utilities, robust statistical functions, data management, lookup, and search functions, as well as powerful chart creation tools. The primary goal in the development of the ENGR117 was to incorporate the use of advanced spreadsheet techniques for their application to engineering problem solving. One example of this is through the use of generalized interpolation (e.g., linear or cubic) functions. Spreadsheet Lookup functions can be combined in such a way as to determine the appropriate bounds for an interpolation operation as well as calculation of intermediate values. Students gain proficiency in the combination (nesting) of complex formulas to perform categorical and criteria-based searches through data. Further, they also gain experience with creating user-defined spreadsheet functions in, for example, the Visual Basic for Applications (VBA) script-editor, which is part of MS Excel.

Numerical programming

As the focus of introductory programming courses such as GMU's CS112 has moved towards object-oriented design and software engineering, the attention paid to traditional numerical programming has diminished greatly. While modern engineering computation environments such as Matlab automate many numerical programming techniques, there remains a need for core competency in skills such as the solving of linear systems, numerical differentiation, and numerical integration. ENGR117 covers these topics in the context of introductory programming. Students learn fundamental data types, algorithms, and linear algebra functions for performing numerical analysis.

3D modeling and scripting

The last component in the development of ENGR117 is lectures and practicums in 3D-based design software. The CE curriculum already contains course work in CAD and geometrics. It lacks, however, any treatment of native 3D design, which is emerging as the standard design medium for building engineering and construction. ENGR117 focuses on Google Skethcup, a free-ware 3D modeling utility that allows students to develop full 3D models of engineering components in an easy to use graphical design environment. The software includes an Application Programming Interface (API) to the scripting language Ruby. Students learn to not only "draw" 3D models, but how to script behavior for those models. One example is an exercise in which the class creates a model of a building project then writes Ruby code to access the physical properties (dimensions) of the building components thereby performing a script-based quantity take-off to support construction estimating tasks. The combination of scripting with 3D modeling reinforces the topics presented in the class.

Subsequent Civil Engineering Computing Course Development

We are planning to incorporate additional computational modules in subsequent courses in our department. This will help students to apply the basic skills taught in ENGR 117 and master their computational skills as they progress towards the completion of their degree. As a start, we are planning to implement NUMerical application problems in Statics and Mechanics of Materials (NumSMM). The following two Matlab computational modules are presented here for illustration purpose.

NumSMM Module 1

After students are familiar with the principle of adding multiple vectors, they will utilize a Matlab tool with graphical user capabilities. The code will be provided and students can alter and modify it depending on the type of inputs and output parameters. The results include both graphical results and a numerical result. This will help students to visualize the magnitude and direction of resultant forces. Figures 1-3 summarize the prototype NumSMM module GUI implemented in Matlab.

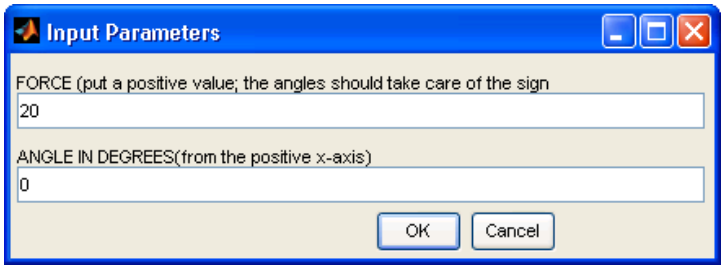

Figure 1. Prompt window for input vector parameters

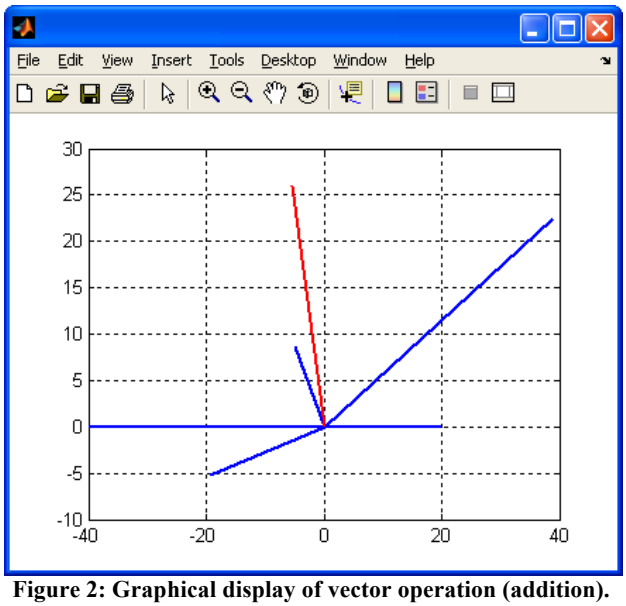

Figure 2: Graphical display of vector operation (addition).

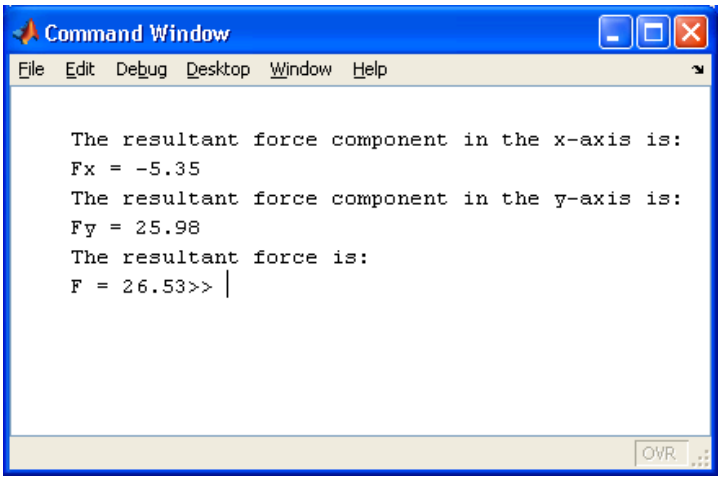

Figure 3: Textual output of vector addition operation.

NumSMM Module 2

After students are taught the principles of truss analysis by the method of joint and method of sections, they will be required to apply their computational skill and determine truss member forces. The graphical user interface in Figure 4 will allow the students to choose boundary conditions, number of joints and number of members for students taking statics. The same module can be used in teaching mechanics of materials because it allows material property inputs. This module will help students to easily visualize the flow of stresses (compression or tension).

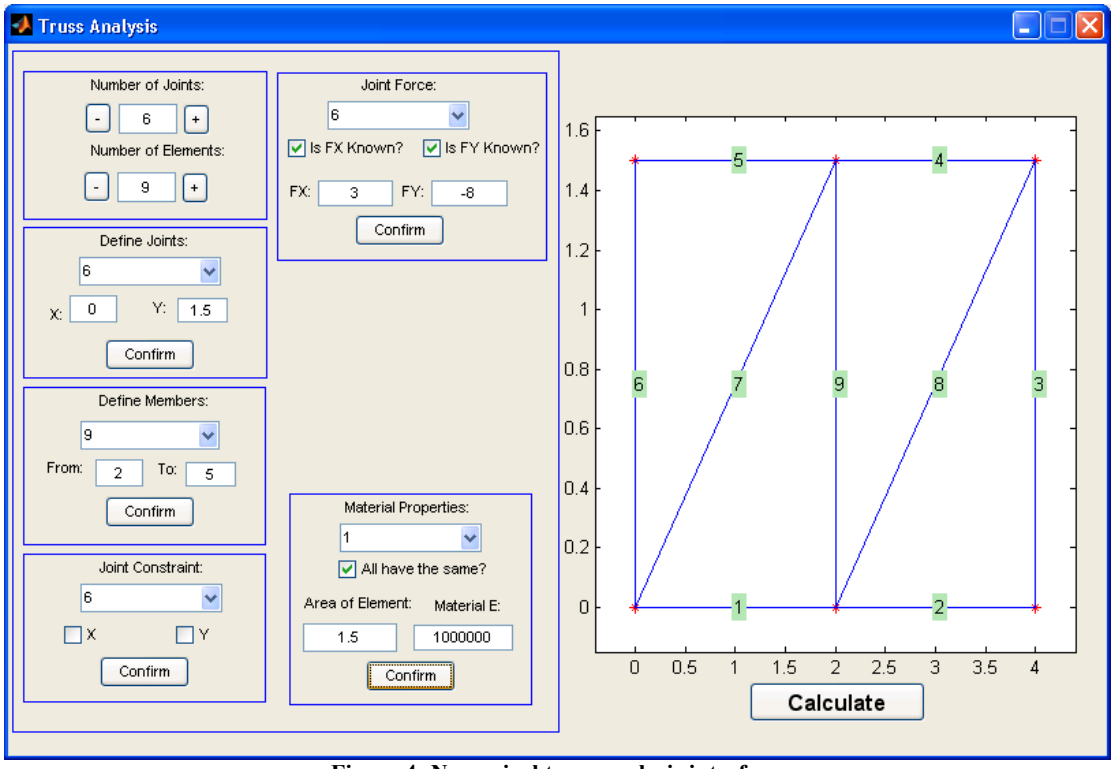

Figure 4: Numerical truss analysis interface

These two modules are just one example of the reinforcement of computational techniques planned in our curriculum in the future. Although implemented in Matlab, students should feel comfortable applying the principles to build their own tools in either spreadsheets of other scripting languages.

Students are already using Google Sketchup for visualization and modeling of design projects in their senior level classes. We anticipate a growing trend of coherence in the adoption and reinforcement of computational skills and knowledge throughout the curriculum.

Assessment

As with any structured curricular change, it is wise to implement a means of assessment for that change. Our chosen initial assessment tool will be the analysis of the performance of our students in the computing portion of the FE examination. In addition to pass rates, categorical (i.e., computing question) statistics are provided from which we can compare student performance for those who took the previous CS112 course and for those who took the new ENGR117 course. We will also track the initial (baseline) performance of the first cohort of students to complete ENGR117 in addition to the reinforcing numerical computing modules to be implemented in our statics, mechanics of materials, and other subsequent engineering courses. We anticipate improvement above baseline. Our long-term assessment goals will be a survey of recent alumni to report on their perceived value of the computing knowledge and skills learned as an undergraduate, applied in professional practice.

Conclusion

This purpose of this paper was to re-examine the core computational skill set of civil engineering undergraduates from the perspectives of accreditation requirements, needs of the practicing engineering and construction industry, and attitudes of the current generation of undergraduates towards computing. ABET program criteria and outcomes were cited as guides for the development of computing curricula, however no absolute program criteria are available related to computing. Rather, the computing applications and skills for inclusion in the curriculum are expected to be commensurate with "modern engineering practice."

This presents a dilemma for civil engineering educators who strive to seek a balance between providing education, which is focused on structured problem solving and application of physical theories to human needs and problems, with training, which is focused on the operation of specific tools. The balance point has long been the division between engineers (those who comprehend the theory and apply it) and technicians (those who implement the designs of others). The trend towards providing newer and more specific software-based curriculum support does now and has for a long time risk moving the focus away from education and towards training. There will always be a need for well-educated engineers to be able to return to "first principles" in order to formulate, solve, and assess the results of modern engineering problems.

Bibliography

1. Abudayyeh, O., Cai, H., Fenves, S.J., Law, K., O'Neill, R., and Rasdorf, W. (2004), Assessment of the Computing Component of Civil Engineering Education, Journal of Computing in Civil Engineering, Volume 18, Issue 3, pp. 187-195
2. Accreditation Board for Engineering and Technology (ABET) (2008), Criteria for Accrediting Engineering Programs 2009-2010, *http://www.abet.org/Linked%20Documents-UPDATE/Criteria%20and%20PP/E001%2008-09%20EAC%20Criteria%2012-04-07.pdf*
3. Garrett Jr., J.H., Flood, I., Smith, I.F. and Soibelman, L. (2004), Information Technology in Civil Engineering - Future Trends, Journal of Computing in Civil Engineering, Vol. 18 Issue 3, p185-186
4. Grigg, N.S., Criswell, M.E., Fontane, D.G., Saito, L., Siller, T.J., and Sunada, D.K. (2004), Integrated civil engineering curriculum: Five-year review, Journal of Professional Issues in Engineering Education and Practice, 130(3), 160-165.
5. El-Zein, A., Langrish, T. and Balaam, N. (2007), A self-practice online tool for teaching and learning computational skills in engineering curricula, International conference on Engineering Education, Portugal
6. Casey, M. (2008) "Work in Progress: How Building Informational Modeling May Unify IT in the Civil Engineering Curriculum." *Proceedings of 2008 ASEE/IEEE Frontiers in Education Conference*, Saratoga Springs, NY.
7. Kent, P. and Noss, R. (2002), The Mathematical Components of Engineering Expertise: The Relationship Between Doing and Understanding Mathematics, Second Annual Symposium on Engineering Education, London
8. Mailhot, P. (2008), Rethinking Engineering Education, Journal of Engineering Education, Vol. 97, pp. 243-244