Computer-Based Skills in an MET Curriculum

William E. Howard and Joseph C. Musto
Milwaukee School of Engineering

Abstract

The TC2K criteria of ABET accreditation for engineering technology programs has allowed for greater flexibility in many areas of curriculum content. Previous requirements included the stipulation that at least one computer language be taught in a BS program, followed by experience using programming skills in technical courses. In the TC2K requirements, a program outcome specifies that students must have "mastery…of the modern tools of their disciplines."

There are a number of ways to meet the TC2K requirement. Software instruction can use high-level programming languages, such as C++ or FORTRAN, spreadsheets, or mathematical computational tools such as MATLAB, Mathematica, MATHCAD, or Maple. In recent years, the distinction between programming language and computational tool has become less clear, as several of the mathematical computational programs also contain powerful programming features.

In this paper, the authors present an overview of computer usage in mechanical engineering technology (MET) programs. The authors also identify several computer-based skills that we think are appropriate for engineering technology graduates, and describe the approach adopted at Milwaukee School of Engineering (MSOE), which includes:

- A balance of MATLAB programming and spreadsheet usage.
- An emphasis on selecting the appropriate tool for a specific task.
- Instruction in the presentation of problem statements and results.
- A focus on applications.

Background

Computer programming has been a required skill in most engineering and engineering technology programs for several decades. From the 1960's through the 1980's, some knowledge of programming was necessary or at least preferred in order to perform computing tasks on mainframe computers. Until very recently, ABET requirements for engineering technology mandated the instruction of at least one computer language. Criterion I.C.6 of the 2003-2004 conventional criteria reads:

Engineering technicians and technologists are dependent upon the computer to effectively perform their job functions. It is therefore essential that students acquire a working knowledge of computer usage. Instruction in applications of software for solving technical problems and student practice within appropriate technical courses is required for all programs. Additionally in Baccalaureate degree programs, instruction must be included in one or more of the computer languages commonly used in the practice of engineering technology. Following formal instruction or demonstrated proficiency in computing skills, baccalaureate students should gain experience using programming skills in technical courses to an extent appropriate for the discipline.[1]

This criterion led to much discussion within engineering technology circles as to exactly what software packages meet this requirement as a "language". Do only high-level structured programming languages such as FORTRAN or C++ qualify? Or do computational packages with loop and logic capabilities satisfy the requirement? This discussion has been made irrelevant by the ABET TC2K requirements that are now in place. Under Criterion 2, Program Outcomes, the first of the eleven required outcomes (a-k) specifies that:

An engineering technology program must demonstrate that graduates have:
a. an appropriate mastery of the knowledge, techniques, skills and modern tools of their disciplines…[2]

Under Criterion 4, Program Characteristics, the technical content requirements include:

Technical courses must develop student knowledge and competence in the use of standard design practices, tools, techniques, and computer hardware and software appropriate to the discipline and goals of the program.[2]

These new requirements give more flexibility to schools to decide what the "appropriate tools" for their programs are. Since ABET's EC2000 criteria for engineering programs has been in place for several years now, engineering programs have also been debating what tools to use.

Gottfried[3] questioned the value of teaching computer programming to first-year engineering students. He argues that most practicing engineers no longer need to write their own programs to solve the engineering problems that they face on a daily basis. As an alternative, he suggests that a course in basic computer skills, with use of a spreadsheet and an equation solver, be a required first-year course, while programming be taught later to those who need it or want to take it as an elective.

Clough, et al.[4], and Collura, et al.[5], present the details of course designed with the Excel spreadsheet and its associated VBA (Visual Basic for Applications) programming environment. This approach has several points in its favor, not of least of which is the widespread availability of Excel. Since almost all students now have access to their own computers, the problem of finding an open lab computer is removed as an obstacle to completing programming assignments. Another advantage of the Excel/VBA solution is that inputs, intermediate calculations, and results are easily viewed and manipulated. While this can become cumbersome for large programs, it is not usually a factor for the types of problems assigned to students in introductory programming classes.

Computing in MET Programs

In order to gain an understanding of what tools are currently used in MET and why, the authors conducted a survey through the Engineering Technology Listserv[6], an electronic communications tool for the ET community administered by Walter W. Buchanan of Northeastern University. Thirty-four responses were received from 27 different MET programs. The results are summarized here:

1. Do you provide formal instruction in any of these computational tools in your BS MET Curriculum? (Please check all that apply)

Reponses from 27 institutions: (Percentages do not add to 100%, since multiple answers were allowed.)

Tool	*# Responses*	*% of Responses*
Excel	21	78%
Visual Basic	12	44%
C++	9	33%
MathCad	8	30%
MATLAB	7	26%
Mathematica	3	11%
Java	2	7%
Maple	0	0
Other	5	19%

"Other" included C (2 institutions), TK Solver, Engineering Equation Solver, and student choice of C++, Java, or Visual Basic. Of the 21 institutions listing Excel, all but one listed another tool, as well.
References to analysis or CAD software were not included in these tallies.

2. What do you consider the MOST IMPORTANT goal of teaching computational software in you program?

Responses from 34 individuals at 27 institutions: (If multiple answers were selected, then the responses were "split" so that the choices of each respondent summed to one. Percentages add to 100% except for rounding error.)

Goal	*# Responses*	*% of Responses*
Students learn computing concepts that can be applied to other tools	11.6	34%
Students can use this particular tool in future classes	10.3	30%
Students learn a formal method of problem solving	9.6	28%
Students learn proper documentation of a solution	1.3	4%
Other	1.3	4%

"Other" included "Use the tool in BOTH classes and on the job", and "Familiarity with use of up-to-date tools. Reinforcement of theoretical concepts taught".

While Excel can be considered to be a universally accepted tool in MET programs (even where it is not taught formally, Excel is probably used by students), use of other tools varies greatly. It should be noted here that Visual Basic usage can include writing and editing Excel macros and creating Excel functions. Therefore, the survey shows no clear consensus on the programming language of choice. As to the most important goal of teaching computational software, respondents viewed learning computing concepts, learning the particular tool for later use, and learning a formal method of problem solving as nearly equal in importance. Although learning proper documentation is probably a goal in most computing courses, it was seen as less important than the three goals previously mentioned.

Computing in the MET Program at MSOE

At MSOE, the MET program had for many years required a computer science class in the freshman year. In 1999, C++ replaced FORTRAN as the language of the required class. In alumni surveys, students rated their ability to use computer software as a tool for problem solving and the ability to use CAD effectively as their two weakest areas. This was a somewhat surprising finding in that CAD, computer programming, and finite element analysis courses were all required in the curriculum. In spite of these computer-intensive courses, it seems that students viewed these courses as isolated from the rest of the curriculum. When the MET Program underwent a major revision in 2002, computer usage was a major consideration.

The Mechanical Engineering Department at MSOE houses both ME and MET programs, and many department faculty members teach in both programs. This has allowed many curriculum and course improvements made in one program to be adapted for the other. In the area of computing, the ME program had also seen student difficulties in a C++ course (the same course taken by MET students) and addressed the problem by creating a freshman-level computing class utilizing MATLAB and Excel, ME-190. The students were still required to take the C++ class, but became familiar with programming basics in the ME-190 class.[7] A second freshman-level course, ME-191, was later added, with MATLAB used to create programs to drive simple hardware from the students' PC's.[8]

While the ME Program experiences have been highly successful (based on faculty observations that students' computing skills are noticeably better), the courses developed could not be duplicated exactly in the MET program. The biggest difference is that ME students are in their second quarter of calculus when taking ME-190. Since MET students are primarily transfer students, most have had no calculus before enrolling at MSOE. The MET faculty desired to have students take a computing course as soon as possible after enrolling, as the enhanced computing skills developed in the course were seen as potentially beneficial in almost all other courses in the curriculum. Therefore, the computing class was to be developed with the assumption that students were at the pre-calculus level of math. The new three-credit class replaced the required C++ class, so the total number of credit hours in the curriculum was unchanged.

The MET faculty chose MATLAB and Excel as the tools for the new class. These tools were selected based on the following considerations:

- Excel is a widely-used, powerful tool available on every lab and student computer[*] on campus.
- Students are familiar with and comfortable with Excel, even if they are unfamiliar with most of the advanced features of the program.
- The MATLAB programming environment includes loop and logic statements typical to those of other programming languages, with minimal program-specific nomenclature.
- The selection of MATLAB complements the use of the Simulink simulation software in later feedback controls and instrumentation classes.
- The use of MATLAB in both the ME and MET curriculum allows more flexibility for faculty members to teach in both programs.

The new MET computing class, MT-3901, was taught for the first time during the 2002-2003. As the class has been taught and revised, the desired outcomes have remained the same. These outcomes are:

- The ability to perform complex calculations.
- The ability to use loops and logic in simple programs.
- The ability to create plots.
- The ability to pose a problem and to present the solution in a clear, concise manner.
- The ability to create a flowchart to describe an algorithm.
- An understanding of the differences between "exact" analytical and numerical solutions.

[*] MSOE has had a laptop computer lease program since 1999. Full-time students are required to participate in this program. Since most MET students are part-time, they were not affected. In 2003, the program was expanded to part-time students. If enrolled in a course designated as "laptop required," they can rent an older computer for the quarter for a small fee.

The pedagogy of the course has included:

- A balance of MATLAB and Excel usage.
- An emphasis on selecting the appropriate tool for a specific task.
- Instruction in the presentation of problem statements and results.
- A focus on applications.

The most significant change in the course since its inception is in how and when MATLAB topics are presented. Most textbooks on MATLAB assume that students have a good knowledge of vectors and arrays prior to learning MATLAB. Also, many texts move into MATLAB "shortcuts" right away instead of presenting a more structured approach. A good example of this is the "implied loop" structure. For example, suppose that a problem involves the value of an account that originally contains $10,000 and receives 5% interest, compounded annually. This problem can be solved in MATLAB with these commands:

```
Yr = [1:20]
Amount = 10000*1.05.^Yr
```

The values of "Amount" can then be plotted against the number of years, as they are both 1 X 20 arrays.

Now consider the following alternative solution:

```
for i = 1:20
Yr(i) = i
Amount(i) = 10000*1.05^Yr(i)
end
```

In the first solution, "Yr" is defined as a 1 X 20 array, and "Amount" is defined as an array by the fact that an operation is performed using the array "Yr." The dot is required in the operation (.^) to indicate that the operation applies to each *element* of the array, not the array itself. This type of solution is very confusing to students.

On the other hand, the second solution requires the students to identify which of the variables are arrays, and to define the values of the arrays through a loop. Not only is the structure easier for the student to understand, the form is similar to that seen in other programming languages. While an experienced user would prefer the first solution in that it is more compact, the wisdom of presenting it to beginning students is questionable.

The approach used in the MSOE course is similar to the "object scaffolding" applied to MATLAB instruction by Sticklen et al.[9] In this approach, students build onto existing knowledge a step at a time. For example, instruction in MATLAB begins with scalar computations, building on the students' knowledge of making scalar computations with a calculator or with Excel. The next step is to introduce vector computations, and then extend to multi-dimensional arrays. As Sticklen et al. point out, this approach seems straightforward, but most MATLAB texts combine these steps into one.

This general concept of building on students' existing knowledge is also applied to the integration of Excel and MATLAB. For example, when the topic of numerical integration is taught, students first calculate the area under a curve using a fixed number of increments in Excel. The problem is then repeated in MATLAB, with the same number of increments as in the Excel solution. Next, the MATLAB solution is modified so that the number of increments can be easily changed, and the students can experience the important step of *converging* to a solution. Finally, the MATLAB code can be modified so that the number of increments is changed automatically until a desired convergence tolerance is realized.
The course topics that are now covered in the 10-week quarter are:

Week 1:	Introduction to Computing, MATLAB Calculations, Vectors
Week 2:	Arrays and matrices
Week 3:	Plotting
Week 4:	MATLAB Programming: Loops
Week 5:	Structured Logic
Week 6:	MATLAB Programming: Nested Loops
Week 7:	Application: Simultaneous equations
Week 8:	Application: Root finding
Week 9:	Application: Numerical Integration
Week 10:	Application: Optimization

There are two class sessions per week: a two-hour lecture session and a two-hour lab session. Note that there are no class sessions devoted exclusively to Excel; students are familiar with entering data and formulas in Excel and advanced features are introduced as needed (matrix operations in Week 2, if statements in Week 5, etc.). The textbook used in the course is *Introduction to MATLAB 6*, by Etter and Kunicky,[10] supplemented by a large number of instructor-written tutorials.

Most of the assignments of this course require very short programs (most less than 20 lines of MATLAB code). These assignments represent good opportunities to emphasize other important skills. Students are required to present their computations in clear and concise write-ups. The use of flowcharts is required in several of the assignments. Although flowcharts may be seen as unnecessary as a planning tool for such short programs, they are valuable for explaining the logic used in a program. Flowcharts are also useful in project planning and in describing processes, so some instruction in their preparation and use is justified. For some assignments, no specific computer tool is specified. In addition to solving the problem, students are expected to explain why they chose to use Excel or MATLAB for their solutions.

Assessment of Curriculum and Course Objectives

It is difficult to make definitive measures of the effect of the described course on the curricular goal of improving student skills in computing. Since most MET students at MSOE are part-time, it can take several years for changes in the curriculum to be reflected in senior and alumni surveys. Qualitative observations of faculty are that students are more comfortable and competent in applying computer tools in other classes, probably due not only to this class but to an increased emphasis on computer usage throughout the curriculum.

Course-level assessment is done for every MET course through student questionnaires and faculty reports completed at the end of the class. During the 2003-2004 academic year, seven students completed MT-3901. Their responses to six standard assessment measures are summarized here. Possible answers for each of the first five statements range from 7, strongly agree, to 1, strongly disagree, with 4 representing a neutral response.

1. *I feel that the courses taken previously as prerequisites prepared me well for this class.*
   Average response = 4.1 (average of all MET classes = 5.5)
   There are no formal prerequisites for this course. New students have been encouraged to take this class as soon as possible so that the skills learned can be utilized in as many other classes as possible. Some new students have taken this class before completing trigonometry and algebra classes; these students may not have been fully prepared for MT-3901. These math classes may be added as prerequisites in the future.

2. *I believe that the course content was consistent with the number of credit hours.*
   Average response = 5.7 (average of all MET classes = 5.6)
   MT-3901 is a three-credit class (two hours lecture and two hours lab per week). Students find the work load for the class in line with other classes in the curriculum.

3. *The textbook was a valuable resource for this class.*
   Average response = 3.3 (average of all MET classes = 4.6)
   The approach taken in teaching this class has necessitated the use of instructor-prepared handouts. The test is used mostly as a reference for MATLAB. Prior experience has shown that if a text is not used much formally in a course, the text rating will be low (but if no text is required for a computer course, students will say that a book is needed).

4. *I can see the relationship of this course to the others in the MET curriculum.*
   Average = 6.0 (average of all MET classes = 5.9)
   Students recognize that computer skills will be important in many of their future classes.

5. *I believe that this course contributed toward my career objectives.*
   Average = 5.9 (average of all MET classes = 5.7)
   Most MET students are working adults who clearly understand the importance of computing skills in the workplace.

6. *My interest in this subject area was sustained or enhanced by taking this course.*
   (Possible responses are Yes and No only)
   All seven students answered Yes (average of all MET classes = 85% Yes)

Overall, student impressions of the new course have been positive. In the next couple of years, data should be available to evaluate the effect of this course on success in subsequent courses.

Conclusions

The TC2K requirements of ABET allow engineering technology programs increased flexibility in incorporating computing skills into their curricula. While knowledge of a programming language is not specifically mandated, many faculty members believe that learning to write computer programs is a skill that is desirable for engineering technology students. The course outlined in this paper is an attempt to develop basic programming proficiency while building computational skills that can be applied in later classes.

Bibliography:

1. *Criteria For Accrediting Engineering Technology Programs; Effective for Evaluations During the 2003-2004 Accreditation Cycle*, Technology Accreditation Commission, Accreditation Board for Engineering and Technology, Inc.
2. *Criteria For Accrediting Engineering Technology Programs; Effective for Evaluations During the 2004-2005 Accreditation Cycle*, Technology Accreditation Commission, ABET, Inc.
3. Gottfried, Byron S., “Should Computer Programming Be Taught to All First-Year Engineering Students?” *1996 ASEE Annual Conference Proceedings.*
4. Clough, David E., Chapra, Steven C., and Huvard, Gary S., “A Change in Approach to Engineering Computing for Freshmen – Similar Directions at Three Dissimilar Institutions,” *Proceedings of the 2001 ASEE Annual Conference and Exposition*.
5. Collura, Michael A., Aliane, Bouzid, Daniels, Samuel, and Nocito-Gobol, Jean, “Learning the Methods of Engineering Analysis Using Case Studies, Excel, and VBA – Course Design,” *Proceedings of the 2004 ASEE Annual Conference and Exposition*.
6. Engineering Technology Listserv, Walter Buchanan, Administrator, http://www.coe.neu.edu/Depts/SET/set/listserv.html
7. Musto, Joseph C. and Howard, William E., “Integration of Laptop Computers into a Freshman Mechanical Engineering Curriculum,” *Proceedings of the 2001 ASEE Annual Conference and Exposition*.
8. Musto, Joseph C., Lumkes, John H. Jr., and Carnell, William, “A Freshmen Programming Course for Mechanical Engineers Using Mechatronics Applications,” *Proceedings of the 2004 ASEE Annual Conference and Exposition*.

9. Sticklen, Jon, Amey, Marilyn, Eskil, Taner, Hinds, Timothy, and Urban-Lurain, Mark, "Application of Object-Centered Scaffolding to Introductory MATLAB," *Proceedings of the 2004 ASEE Annual Conference and Exposition*.
10. Etter, Delores, Kuncicky, David, with Hull, Doug, *Introduction to MATLAB 6*, 2nd Edition, Pearson Prentice Hall, 2004.

WILLIAM E. HOWARD
Ed Howard is an Associate Professor in the Mechanical Engineering Department and Program Director of the Mechanical Engineering Technology Program at Milwaukee School of Engineering. He holds a B.S. in Civil Engineering and an M.S. in Engineering Mechanics from Virginia Tech, and a PhD in Mechanical Engineering from Marquette University. He has 14 years of industrial experience, mostly in the design and analysis of composite structures. He is a registered Professional Engineer in the state of Wisconsin.

JOSEPH C. MUSTO
Joe Musto is an Associate Professor and Mechanical Engineering Program Director at Milwaukee School of Engineering. He holds a B.S. degree from Clarkson University (Potsdam, NY), and an M.Eng. and Ph.D. from Rensselaer Polytechnic Institute (Troy, NY), all in Mechanical Engineering. His industrial experience includes engineering positions with Eastman Kodak Company (Rochester, NY) and Brady Corporation (Milwaukee, WI). He is a registered Professional Engineer in the state of Wisconsin.