A Change in Approach to Engineering Computing for Freshmen – Similar Directions at Three Dissimilar Institutions

David E. Clough, Steven C. Chapra, and Gary S. Huvard
University of Colorado / Tufts University / Virginia Commonwealth University

Abstract

Introductory computing courses for engineering students at Tufts University, the University of Colorado, and Virginia Commonwealth University (VCU) have undergone revision and development over the past year. Although the scope of these courses differs among the three institutions, similar threadlines have emerged. These include emphases on engineering problem solving, elementary numerical methods, and algorithmic programming. Software vehicles include Mathcad, Matlab, and, in particular, Excel and its VBA programming language. Use of a traditional, stand-alone programming language, such as Fortran or C/C++, is postponed beyond these introductory courses. There are strong, relevant pedagogical and practical bases for this common approach and results from initial course offerings are most promising.

Background

The teaching of introductory computing at the freshman level has long been fraught with controversy and emotion, possibly far more than deserved. Those most opinionated are often the most out of touch: they don't do much computing themselves; they are isolated from the day-to-day computing activities of engineering professionals; and/or they are unfamiliar with the teaching of 17-year-olds just out of high school. A rational approach to introductory computing is based on the real needs of students and professionals. These should be assessed through survey, study and evaluation, and then used as the basis for curriculum design. We believe we have done this.

There have been tendencies across the US to go in one of two directions when it comes to introductory computing for engineering students:

1) The *"tools"* approach. Here, the focus is on the built-in capabilities of a number of software packages. Students solve a variety of engineering problems within the confines of the software's menu options. The features of the software define the scope of the problem solving. Engineering faculty usually teach this course.

2) The *"CS101"* approach. This is an introductory course in computer programming, most often taught as a service course by the computer science department. From the late 60's to the early 90's, Fortran was usually the language of choice for engineering students, although Pascal was used for a number of years. The current trend for this course is to use the C/C++ language as a vehicle, a choice apparently driven by the GUI requirements of personal

computer applications. The fundamentals of programming are emphasized at the expense of engineering problem solving. Most engineering students never again write a C/C++ program after the course is completed.

Common basis and rationale for the new courses

Our approach is middle-of-the-road when compared to the above. We focus on engineering problem solving:

- interpretation of the problem statement
- design of the solution strategy
- getting the results, usually numerical
- interpretation and checking of the results
- display and documentation

Yet we retain the teaching of programming fundamentals:

- data structure
- algorithm structure
- modular design

And we require that students both develop skills and attain knowledge in these areas.

The vehicles we use are software packages that are extensible through programming:

- Microsoft Excel 2000 with Visual Basic for Applications (VBA)
- Mathcad 2000
- Matlab 6

We teach programming fundamentals with VBA as a natural extension of Excel's problem-solving environment. Since VBA is a fully-featured language, including object-oriented capabilities, it provides an excellent environment for knowledge development in programming, knowledge that is portable to other environments, first to Mathcad and Matlab, and later to other free-standing languages, such as C/C++ and Fortran. Also, there is a high degree of computer ownership among our students, and Excel/VBA is already installed on virtually all of these machines. There are low-cost student versions of Mathcad and Matlab, and our instructional labs have both installed.

Our students continue to use these tools in their subsequent courses in engineering and science. Thereby, they have the possibility to maintain their programming knowledge, something that was missing from earlier approaches using programming languages, such as Fortran and C/C++. We want to avoid the "soon forgotten" syndrome. And, through knowledge of programming, students are able to extend the capabilities of the software packages. In being able to do so, they differentiate themselves from other students and professionals who are limited to using built-in features. From a broader viewpoint, we recognize the discipline of computer programming for its intellectual content. This transcends immediate utility and the development of specific skills.

The knowledge embodied in structured programming is of fundamental value to engineers and scientists and provides value in innumerable and unseen ways.

We recognize that, in today's computing world, a freshman introductory course can only lay a foundation and open doors. We expect that those of our students who have a strong interest in computing will take additional courses in computer science, programming, numerical methods and computer engineering. And students who have had our similar courses in the past enjoy more success in these following courses than those who have not. But, at the same time, we believe that a well-engineered, effective introductory course sets the right direction early on for all students. And, given the level of preparation of our entering students in problem solving, the experience provided by our course is essential to survival in engineering.

Our instructional *m.o.* has been to decrease the amount of time spent in the traditional classroom lecture setting and increase hands-on time in the computer laboratory with expert coaches nearby to help. These coaches are most often undergraduate assistants who were successful in the course earlier and have excellent mentoring talents.

Settings for the courses

Tufts University: Introduction to Computers in Engineering [EN-1]

Steve Chapra developed and taught this course first in the Fall 1999 semester. It is a 1-1/2-credit-hour course, and thus must be more limited in scope than a standard 3-credit-hour course. This course is taught in a single section to all engineering and computer science freshmen at Tufts, about 250 students each fall semester. Lab sections are about 20 students each.

After an introduction to computer systems, email and the Internet, there is a brief segment on spreadsheet problem solving with Excel. The bulk of the course teaches programming fundamentals using Excel and VBA. Students develop a major Excel/VBA project and make a PowerPoint-based presentation that is videotaped and critiqued.

The contact time allocated to the course does not allow for inclusion of other software packages, such as Mathcad or Matlab.

University of Colorado: Introduction to Engineering Computing [GEEN 1300]

This 3-credit-hour course has been taught at Colorado since 1986 to all engineering students, except EE's. It is not taught to computer science students. The course was based on the Fortran language with some content covering Excel and Matlab, but Dave Clough revised the course for the Fall 2000 semester to use VBA instead of Fortran. It is taught to about 250 students each academic year in sections of 75. Lab sections are about 18 students each.

The course begins with a 4-week segment on engineering problem solving with Mathcad as the software vehicle. This is followed by a 4-week segment on spreadsheet problem solving with Excel. Programming fundamentals are then taught in a 4-week segment using Excel and VBA. In a final 4-week segment, vector/matrix calculations are introduced using Matlab, and the programming techniques learned in VBA are ported to Matlab's m-script language. Elementary numerical methods introduced include root finding, fixed-point iteration, curve fitting, and Gaussian elimination.

Virginia Commonwealth University: Computer Methods in Engineering [ENGR 115]

This introductory course is the first element in a school-wide initiative to teach all engineering students a consistent set of fundamental computing skills. This 1-credit-hour course will be piloted by Gary Huvard with chemical engineering freshmen in bi-weekly, 1.5-hour meetings in a computer classroom in Spring 2001 and implemented for all engineering freshmen in Spring 2002. Problem selection will be integrated with a concurrent course on statics and strength of materials.

The course plan includes 2 weeks on Web page creation using HTML. This is followed by the major, 10-week segment on Excel and VBA. Of this, five weeks will be devoted to writing VBA functions and subroutines. A final 3-week segment introduces the use of Kaleidagraph for data analysis and the creation of publication quality graphs.

For chemical engineers, the freshman course will set the stage for teaching advanced problem-solving and programming skills using Excel, VBA, and Mathcad in the sophomore mass and energy balances course and beyond. Mechanical and Electrical Engineering will introduce advanced skills using Mathcad and Matlab, respectively, through the sophomore and junior year. Computer Science courses on programming languages (C/C++, Fortran etc.) will be encouraged as technical electives and simulation software is taught later in the curriculum as well. However, the emphasis will be to reinforce skills first learned in the freshman year through continued use, in <u>all</u> courses, of a consistent set of computing and programming tools through all four years of the curriculum. VCU has purposely chosen the software packages that they feel most students will continue to use after graduating to industrial positions.

Pedagogical approach – an example

The combination of Excel and its companion programming language provides an excellent setting for the teaching of elementary numerical methods and programming fundamentals. Students can prototype a numerical method, such as bisection root finding, on an Excel spreadsheet. In this way, the algorithm and its results are spread out in front of the student via the live spreadsheet calculations. See Figure 1 below. In this example, the depth of liquid in a spherical tank is solved, given the liquid volume and the inside radius of the tank. The actual equation solved is

$$f(h) = h^3 - 3Rh^2 + 3V/\pi$$

where h: liquid level, m
R: tank inside radius, m
V: liquid volume, m^3

	A	B	C	D	E	F	G
1	Example Spherical Tank						
2							
3	Radius	10	ft		Tank capacity	118.6	m^3
4		3.048	m			118613	L
5	Liq volume	1000	gal			4189	ft^3
6		3.79	m^3			31332	gal
7	Depth	0.652	m				
8		2.14	ft				
9	Bisection method of live solution						
10	Iteration	h_1	$f(h_1)$	h_2	$f(h_2)$	h_{mid}	$f(h_{mid})$
11	1	0	3.615	6.096	-109.653	3.048	-53.019
12	2	0	3.615	3.048	-53.019	1.524	-14.083
13	3	0	3.615	1.524	-14.083	0.762	-1.252
14	4	0	3.615	0.762	-1.252	0.381	2.343
15	5	0.381	2.343	0.762	-1.252	0.572	0.815
16	6	0.572	0.815	0.762	-1.252	0.667	-0.154
17	7	0.572	0.815	0.667	-0.154	0.619	0.347
18	8	0.619	0.347	0.667	-0.154	0.643	0.101
19	9	0.643	0.101	0.667	-0.154	0.655	-0.026
20	10	0.643	0.101	0.655	-0.026	0.649	0.038
21	11	0.649	0.038	0.655	-0.026	0.652	0.006
22	12	0.652	0.006	0.655	-0.026	0.653	-0.010
23	13	0.652	0.006	0.653	-0.010	0.653	-0.002
24	14	0.652	0.006	0.653	-0.002	0.652	0.002
25	15	0.652	0.002	0.653	-0.002	0.652	0.000
26	16	0.652	0.000	0.653	-0.002	0.653	-0.001
27	17	0.652	0.000	0.653	-0.001	0.652	0.000
28	18	0.652	0.000	0.652	0.000	0.652	0.000
29	19	0.652	0.000	0.652	0.000	0.652	0.000
30	20	0.65245	0.000	0.65246	0.000	0.65245	0.000

Figure 1. Bisection spreadsheet

All formulas are live and bisection algorithm decisions are implemented with Excel's IF functions. The formulas in cells B12 and D12 represent the decisional basis of the algorithm:

```
B12:  =IF(C11*G11>0,F11,B11)
D12:  =IF(E11*G11>0,F11,D11)
```

and the simple numerical method is shown by the midpoint formula in cell F11:

```
F11:  =(B11+D11)/2
```

Although illustrative and educational, the spreadsheet solution is limited. The number of iterations is arbitrary, and there is no convergence control. Also, it is awkward to convert the method for solution of another equation. However, with this prototype built on the Excel spreadsheet, the student is ready to elevate the application into the form of a user-defined function in VBA. This is illustrated in Figure 2 below.

The user-defined function Bisect is conveniently implemented in one formula on the spreadsheet. The function is re-entrant and can be invoked from multiple locations on the spreadsheet. This example includes checking of initial guesses and convergence control. Iteration limits can be added easily.

In the Colorado course, given an illustrative example like this, students then develop similar spreadsheet prototype → VBA function applications, such as Newton's method, false position, Golden Section search, and the Wegstein method.

```
Option Explicit
Function Bisect(x1, x2)
Dim xmid As Single, xold As Single, tol As Single
tol = 0.000001
xold = x1
If f(x1) * f(x2) > 0 Then
  Bisect = "bad initial guesses"
Else
  Do
    xmid = (x1 + x2) / 2
    If Abs((xmid - xold) / (xmid + xold)) < tol Then
      Exit Do
    End If
    xold = xmid
    If f(x1) * f(xmid) > 0 Then
      x1 = xmid
    Else
      x2 = xmid
    End If
  Loop
  Bisect = xmid
End If
End Function
Function f(x)
  Const Pi = 3.1415927
  Dim V As Single, R As Single
  V = Range("Volume")
  R = Range("Radius")
  f = x ^ 3 - 3 * R * x ^ 2 + 3 * V / Pi
End Function
```

C5 = =Bisect(h1val,h2val)

	A	B	C
1	Development of Bisect Function		
2			
3	0	h1	
4	20	h2	
5			4.31 h
6			
7	10	Radius	
8	500	Volume	

Figure 2. Bisect user-defined function in VBA and as implemented on Excel spreadsheet

We have found that the VBA language provides an excellent setting for the teaching of algorithm and data structure. Its algorithm structure is on a par with that of Fortran 90/95 and superior to that of C/C++ and Matlab, especially in the case of loops. The VBA language is more explicit and verbose than C/C++ and, consequently, less intimidating to students with no background in programming. Students also must make use of objects, properties and methods in VBA in order to communicate with the spreadsheet; so, they are introduced to object-oriented programming.

For those students interested in building their programming and software engineering expertise, these introductory courses based on VBA provide a helpful stepping stone to subsequent courses in computer science that improves the students' chances for success and learning in these latter courses.

Student reaction

Since the Tufts course has been taught twice, the Colorado course only once, and the VCU course is about to be offered, student feedback does not have a long track record.

Based on exit surveys of students at Colorado at the end of the Fall 2000 semester, we make the following observations:

- Students preferred Excel/VBA over Mathcad and Matlab. Mathcad finished a distant 2nd with Matlab close behind.
- In Mathcad, students liked the presentation of equations, symbolic operations, and explicit handling of units. They did not like the finicky editing and the graphics.
- In Excel/VBA, students liked the logical layout of the spreadsheet with everything organized and the results visible. Reviews on programming with VBA were mixed but tilted to the positive. Some thought that VBA could get more complicated than they would like. Students wished that Excel had symbolic capabilities.
- As for Matlab, students liked the vector/matrix capabilities, although they realized that much of this power was beyond their appreciation. Students disliked the Matlab syntax that includes "dots" for array operations. They strongly disliked the Matlab command window interface, considering it primitive when compared to Mathcad and Excel. They really liked Matlab's 3D plotting capabilities. They disliked Matlab's C-like loop structures, seeing them as inferior to those of VBA.
- Over 80% of the students used Excel/VBA in one or more other courses during the same semester of the computing course. About 40% of the students used Mathcad in another course. Only about 10% of the students used Matlab outside of the computing course. Such use was generally spontaneous and not required in these other courses. Students appreciated the immediate impact of their learning.
- In comparison to their other freshman-level courses (calculus, chemistry, physics, etc.), students felt strongly that they learned significantly more in the computing course.

Text materials

Since the courses at Tufts, Colorado and VCU represent a non-traditional approach to introductory computing, there is a natural scarcity of text and support materials. Steve Chapra at Tufts used his own draft manuscript of a text on VBA programming[1] and several chapters of Clough's draft manuscript of a text on Excel problem solving[2].

Dave Clough at Colorado used a McGraw-Hill custom text with excerpts from books on engineering problem solving[4], Mathcad[7] and Matlab[6], along with the abovementioned Excel chapters[2]. The so-called "VBA for Dummies"* text by Walkenbach[8] was used for background on VBA programming.

Gary Huvard will begin with Gottfried[5] (Excel), Walkenbach[8] (VBA), and supply materials for segments on web-page design and Kaleidagraph. He expects to supplement these with examples and problems from Clough[2], from Duncan and Reimer[3], and the text used for the freshman statics course at VCU.

We expect that the quality and availability of text and support materials for these courses will improve greatly in the coming year. The authors will make available their own instructional materials to interested parties upon request.

* The "Dummies" part of the title is definitely misleading. This user-friendly book contains a significant amount of material on programming fundamentals in VBA.

Where from here?

This effort is still at a fledgling stage. Initial feedback from students is a marked improvement from that during earlier, more traditional approaches to engineering computing. Their level of motivation and excitement is higher, and this is fueled by their perception of the immediate utility of the course topic material. They are using what they learn in other courses in the same semester. Although time will be the judge of our mid-term and long-term hypotheses, assessment by our upper-division students and alumni indicate that we are on the right track.

Bibliography

1. Chapra, S.C., *Excel 2000 With Visual Basic For Applications For Engineers* , draft manuscript.
2. Clough, D.E., *Spreadsheet Problem-solving for Engineers & Scientists Using Excel 2000* , draft manuscript.
3. Duncan, M. A. and Reimer, J. A., *Chemical Engineering Design and Analysis: An Introduction*, Cambridge University Press, 1998.
4. Eide, A.R., et al., *Engineering Fundamentals and Problem Solving, 3/e* , selected chapters, 100 pp., McGraw-Hill, 1997.
5. Gottfried, B.S., *Spreadsheet Tools for Engineers: Excel 2000 Version*, McGraw-Hill, 2000.
6. Palm, W.J., *Introduction to Matlab for Engineers* , selected chapters, 232 pp., McGraw-Hill, 1999.
7. Pritchard, P.J., *Mathcad: A Tool for Engineering Problem Solving* , selected chapters, 166 pp., McGraw-Hill, 1999.
8. Walkenbach, J., *Excel 2000 Programming for Dummies*, IDG Books, 1999.

DAVID E. CLOUGH

Dave Clough is Professor of Chemical Engineering at the University of Colorado and has been on the faculty there since 1975. His research interests are centered on the optimization and control of chemical processes. He teaches process control, applied statistics, and introductory computing. He lives in a log home in the Rocky Mtns. Email: David.Clough@Colorado.edu

STEVEN C. CHAPRA

Steve Chapra holds the Lewis Berger Chair for Engineering Computing in Civil and Environmental Engineering at Tufts. His research interests are surface water quality modeling, numerical methods and computer applications in environmental engineering. He teaches in these areas and introductory computing. He was drawn to environmental modeling by his love of the outdoors and fascination with computers. Email: Steven.Chapra@Tufts.edu

GARY S. HUVARD

Gary S. Huvard is Associate Professor and Assistant Chair of Chemical Engineering at Virginia Commonwealth University. He joined the VCU faculty in 1999 after 11 years in positions with BFGoodrich and duPont and 10 years in private practice. He currently teaches seven different undergraduate courses including the freshman computer skills course. He doesn't live in a log home. Email: gshuvard@vcu.edu