University of Colorado

There are five goals in teaching computing to undergraduates in chemical engineering:

- fundamental knowledge of computing, programming and computers
- awareness of and preparation in emerging aspects of computing
- computing requirements in the other courses of their curriculum
- knowledge and skills required by engineers in their day-to-day professional lives
- opening the door for further study and specialization in computing and computer science

The argument for fundamental knowledge is sound. Such knowledge in computing will transcend the skills and tools of the day. Fundamentals provide the foundation. A criticism is that fundamentals tend to be abstract and difficult for students to grasp and appreciate. Most students learn better inductively, generalizing fundamentals from specific and practical exercises and examples.

Given the rapid rate of change in computing, teaching only the state of the art in the profession will leave students out of date by the time they get there. Therefore, educating them in emerging trends is important. A problem with this is judging which trends will stick and which will be flashes in the pan.

Students will appreciate and be motivated by the acquisition of skills that they can put to immediate use, even during the semester in which they are taking the introductory computing course. Of course, focusing too much on immediate needs may miss the mark when it comes to professional needs. Many computing tools used in the curriculum satisfy learning objectives but are of little use in professional life.

Teaching students in the context of computing vehicles used by practicing professionals has attractive payouts down the road. Focusing on the day-to-day problem solving activities of engineering professionals has high relevance and importance. However, there are difficulties. The learning curve for some software packages is far too steep, and some packages have a knowledge prerequisite, especially in mathematics, that far outstrips the abilities of first-year students.

There is also a significant problem with using tools that automate tasks, where the user has little view of the internal operations of the task – the *black-box* syndrome. This works fine and is greatly appreciated by the professional that already has knowledge of and experience with the task being performed. But there is a great temptation for mindless button-pushing (or mouse-clicking) by students who should be learning what is behind the button. Also, there is the danger of becoming trapped by the built-in capabilities of one's software, in other words, being incapable of extending the software capabilities through programming.

A few students will want to specialize in computing. For example, some students will become software developers in the context of engineering applications. These

individuals will require more education in computing, perhaps a minor or even a double degree. The introductory computing course in engineering should not attempt to redirect these students away from computer science; rather, it should open the door.

Since valid arguments can be made for the five areas of need listed above, it becomes a challenge to design an introductory computing experience that balances and, at least in part, satisfies the needs. We have attempted to meet that challenge at the University of Colorado.

In engineering at the University of Colorado, introductory computing is taught under an umbrella course number (GEEN 1300 Introduction to Engineering Computing, 3 credit hours). The various engineering degree programs (chemical, mechanical, civil, architectural, environmental, aerospace) each teach a section of this course, and these sections take on "flavors" according to the preferences of the particular program. Students in electrical engineering, computer engineering, and computer science do not take this course, rather a typical "CS101" course based on C/C++. A significant fraction of the entering students, typically 30%, are "open option," not having declared an engineering major. These students are included in the sections of the GEEN 1300 based on their interest in and leanings toward an engineering major. Also, there has occasionally been an additional section of the course for "open option" students and students not yet in the College of Engineering.

Two years ago, ChE at Colorado initiated a change in this course, taking it away from its traditional Fortran/Excel base. In this transition, two central themes were preserved: scientific/engineering problem solving and structured programming. The new course is divided into four roughly-equal parts. First comes a segment on engineering problem solving using the Mathcad package. This is followed by a segment on engineering problem solving and elementary numerical methods with Excel. The third segment expands the use of Excel by introducing structured programming via its Visual Basic for Applications (VBA) language. And the final segment continues the themes with the Matlab package along with an introduction to vector/matrix calculations. Problems from chemistry, physics, engineering and ChE are used throughout.

There are some pedagogical keys to the success of this course. The combined use of Excel and VBA has a strong supporting case. Providing students with knowledge and skills that they can use immediately, during the same semester, in other courses and activities is important to student motivation. Providing a gateway to subsequent use of the software tools and, for some students, to building their computing knowledge in follow-on courses completes the picture.

The introductory engineering computing course has been established with a set of objectives that include:

1. Problem Solving

- Apply the "engineering method" to the solution of quantitative problems

- Evaluate engineering formulas, carrying units and appropriate precision through calculations
- Practice working in groups to tackle large-scale engineering problems

2. Symbolic Computing

- Enter and edit symbolic expressions in computer software
- Manipulate and solve algebraic expressions
- Carry out symbolic manipulations for calculus

3. Spreadsheet Techniques

- Develop efficient spreadsheet skills
- Set up and interpret "what-if" and case study scenarios
- Organize and layout spreadsheet solutions to engineering problems

4. Programming Fundamentals

- Learn how information is represented by different data types
- Learn program-flow algorithm structure and modularity
- Program with object-oriented features

5. Elementary Numerical and Statistical Methods

- Develop the ability to solve single nonlinear algebraic equations using elementary numerical methods, such as bisection, false position or Newton's method
- Solve sets of linear and nonlinear algebraic equations
- Carry out regression calculations

6. Software Tools

- Develop skills with and knowledge of the following software tools:
  Mathcad 2001
  Excel 2000 & Visual Basic for Applications (VBA)
  Mathlab 6

The "Problem Solving" objective is a carryover from the old "slide rule" courses. Most entering students lack practice and abilities in numeric problem solving. Many of the lessons from the old courses still have much value and prepare students for the activities in their other courses, in particular, the ChE material & energy balances course. Achieving this objective has an obvious beneficial long-term impact.

"Symbolic Computing" is of immediate utility in the students' math courses. They also make use of this in their science and engineering courses. It is common for students to check their manual work using the computer. A byproduct lesson learned is that not all equations or systems of equations have analytical solutions [not obvious to many freshmen].

"Spreadsheet Techniques" provides a problem-solving methodology that has the broadest and longest impact of any objective in the course. Excel is the day-to-day

problem solving tool of most practicing ChE's, and it is the software tool used most frequently by ChE students. Spreadsheet methods are becoming recognized in their own right along with the need to teach them separately to ChE students. The author's AIChE short course in spreadsheet problem-solving has been one of the most frequently offered courses (approaching 100 offerings) over the past dozen years.

"Programming Fundamentals" represents the *lost objective* in many engineering computing courses. It has been retained in this course in a creative way. The fundamentals of structured, algorithmic programming and data structure are introduced via the VBA language within Excel. This provides a natural setting for students to "elevate" prototypes developed on the spreadsheet into more elegant and efficient VBA macros (Subs) and user-defined functions. The portability of the programming concepts is further emphasized by learning the m-script language in Matlab. Achieving this objective opens the door for students in two important ways:

- Students who move on to take the computer science courses have a leg up on students with no background in programming – our students enjoy greater success in these follow-on courses

- Programming opens the door to extension of many software packages – without the ability to program, users are forever limited to the conventional, built-in capabilities of the packages.

There are certain numerical and statistical methods that represent the bread-and-butter of applications in engineering, both in the academic curriculum and in practice. Several of these are within reach of entering students, and these are represented in the "Elementary Numerical and Statistical Methods" objective. Apart from the practical relevance of equation-solving and regression methods, the lessons learned through a disciplined approach to Gaussian elimination are of great general value.

Our students are exposed to a great variety of software tools during their undergraduate ChE curriculum [Word, Powerpoint, Excel, Mathcad, Matlab, Mathematica, Simulink, Polymath, EZ-Solve, HYSYS, Aspen+, Minitab, Control Station, LabView, LadSim, AutoCAD, to name a few]. To achieve the "Software Tools" objective, we choose to expose the students to several software tools that will be of general utility, teach portable concepts, be accessible and easily acquired, be relevant to their other courses and/or to professional practice. Additionally, we need to prepare the students for the onslaught of the other software packages they will encounter. They obviously need to become skilled at picking up new tools quickly.

The course objectives are embodied in a course outline as follows:

Topic	No. of lectures	No. of Labs
Introduction, problem solving, MathCAD	6	3
Spreadsheet problem solving, Excel	7	4
Introduction to programming, VBA	7	4
Numerical Methods, MATLAB	7	4
	27	15

The introductory engineering computing course is taught in several sections that align with the engineering disciplines and the instructors are faculty from those disciplines. Consequently, there is a limited emphasis on examples and problem solving within the particular discipline of the instructor and section. However, the course sections are similar enough that students can cross over sections. This is also important for engineering students who have not declared a specific major yet.

Most believe that a "learn by doing" approach must be an integral part to an introductory computing course. The GEEN 1300 course incorporates a lab component, replacing a 1-hour lecture meeting with a 2-hour workshop in a computer laboratory. The workshops are tutorial in nature with some open-ended, exploratory content. Lab sections are mentored by upper-class undergraduate students who were successful in the course as freshmen. Experience over the years has taught us that this works better than instruction by graduate student TAs, many of whom come to us with varied and limited computing experience.

Outside work is dominated by weekly computing projects that require deliverables of computer files and written reports. There are frequent in-class demonstrations in lieu of conventional lecture. All lecture materials are available to students before class time as Powerpoint files.

Nearly all students have their own computers in their dorm rooms, and, although University computer labs are available in many locations for their use around the clock, students prefer to do their wok on their own computers. For about 50% of the course, this need is easily answered by Excel, a standard package on their computers. Many students elect to acquire Mathcad for a student price of about $125. Fewer choose to buy the student edition of MATLAB, although many do this later in their academic careers when the software package comes into more frequent use.

From our alumni and employer surveys, we find that MathCAD and MATLAB are not generally available to practicing ChE's. Of course, Excel is available to all. So, the former packages answer mainly educational and academic needs.

Learning Goals (GEEN 1300)

1. Problem Solving

- Learn to apply the "engineering method" to the solution of quantitative problems
- Develop the ability to evaluate engineering formulas, carrying units and appropriate precision through calculations

2. Symbolic Computing

- Develop the skills to enter and edit symbolic expressions in computer software
- Learn to use computer software to manipulate and solve algebraic expressions
- Learn to carry out symbolic manipulations for calculus

3. Spreadsheet Techniques

- Develop efficient spreadsheet skills
- Learn to set up and interpret "what-if" and case study scenarios
- Learn to organize and layout spreadsheet solutions to engineering problems

4. Programming Fundamentals

- Learn how information is represented by different data types
- Learn program-flow algorithm structure and modularity
- Learn to use features of object-oriented programming

5. Elementary Numerical and Statistical Methods

- Develop the ability to solve single nonlinear algebraic equations using elementary numerical methods, such as bisection, false position or Newton's method
- Learn to solve sets of linear and nonlinear algebraic equations
- Learn to carry out regression calculations

6. Software Tools

- Develop skills with and knowledge of the following software tools:
  Mathcad 11
  Excel 2002 and Visual Basic for Applications (VBA)
  Matlab 6.5

The engineering computing course in ChE at Colorado introduces students to various bread-and-butter numerical methods. This is done in a "crawl-then-walk-then-run" fashion where students first solve problems with pencil and paper ("crawl"), then use the spreadsheet environment ("walk"), then program the method in VBA and later in MATLAB ("run"), and finally use the "black box" capabilities of the software packages (Mathcad, Excel and MATLAB). This approach has two benefits:

- students gain an appreciation and understanding of the method and its limitations that is not possible by merely pushing the buttons of the "black box" software features, and
- students learn the discipline required to understand and carry out algorithmic numerical method.

Methods taught include equation solving (single nonlinear equations, methods such as bisection, false position and Newton's), solving sets of linear algebraic equations via Gaussian elimination, and linear regression.

Learning Goals for CHEN 4580 – Numerical Methods for Process Simulation

1. Macroscopic Conservation Laws
- Understanding of general conservation laws
- Ability to write unsteady-state conservation laws that include the accumulation term omitted in steady-state problems
- Recognition of ordinary differential equations with straightforward analytical solutions

2. Algebraic Equations
- Ability to solve single non-linear equations
- Understanding of solution methods for systems of non-linear equations and ability to implement those methods in a modern computational environment
- Understanding of how problem structure can be used to improve solution efficiency
- Recognition of common chemical engineering applications leading to algebraic equations

3. Ordinary Differential Equations
- Knowledge and use of single-step and multi-step methods for solution of initial value ordinary differential equation problems
- Understanding of what a "stiff" differential equation is and appreciation for how stiff equations can be handled differently
- Familiarity with available software for solving ordinary differential equations
- Recognition of common chemical engineering applications leading to ordinary differential equations

4. Microscopic Balances
- Ability to use a shell balance to derive a local, differential conservation equation
- Recognition of common chemical engineering applications involving microscopic balances

5. Split Boundary Value Problems
- Understanding and use of numerical methods for split boundary value problems

- Familiarity with available software for split boundary value problems
- Recognition of common chemical engineering applications leading to split boundary value problems

Catalog Description

The use of macroscopic and microscopic balances for development of mathematical models to describe common chemical engineering unit operations; numerical methods for solution of model equations. Prerequisites CHEN 3210, and 3220.

The third course in the computing sequence at U. Colorado is "Applied Data Analysis". The learning goals are given below.

Learning Goals (CHEN 3010)

1. Measurement Fundamentals and Error Analysis
- Understanding of measurement principles including accuracy and precision
- Ability to assess and manage experimental error in engineering calculations
- Knowledge of measurement techniques for common variables, such as pressure, flow and temperature

2. Characteristics of Experimental Data
- knowledge of probability concepts that relate to experimental measurements
- ability to describe measurements using common distributions
- ability to represent distributions using standard graphical and computing techniques

3. Statistical Methods
- understanding of the relationship of sample statistics to background distributions
- ability to compute sample statistics and confidence intervals
- ability to apply hypothesis tests common to engineering analyses

4. Model Building
- understanding of the concepts of regression analysis, including correlation and analysis of residuals
- ability to compute linear, including multilinear and curvilinear, and nonlinear regression
- ability to express confidence intervals on model parameters
- ability to assess goodness of fit and discriminate amongst competing models

5. Design of Experiments
- understanding of factorial design of experiments and response surface methods
- ability to plan an efficient experimental campaign based on factorial design

- understanding of the principles of analysis of variance as they apply to factorial design
- ability to process and interpret the results of factorial experiments
- ability to develop response surface models from the results of factorial experiments and use these models for prediction and evaluation.

Catalog Description

Students learn to analyze and interpret data. Topics include typical engineering measurements, graphical presentation and numerical treatment of data, statistical inference, regression analysis, and design of experiments. Prerequisites are GEEN 1300 and APPM 2360.