"Computing Through the Curriculum: An Integrated Approach for Chemical Engineering"

CACHE Corporation (2/01/04)

Abstract

This white paper focuses on the integration of computing throughout the chemical engineering curriculum. The computing experience for undergraduates in chemical engineering should have continuity and be coordinated from course to course, because a single software solution is difficult to achieve in practice. This paper covers the topics of teaching computer programming, software selection, mathematical modeling instruction, and teaching process and product design. Appendices include a recent survey of computing practices in industry and descriptions of integrated computing approaches at selected departments.

TABLE OF CONTENTS

1.0 Computing and the Systems Approach in Chemical Engineering

2.0 Teaching of Computing Programming and Computer-based Courses
- 2.1 Teaching Computer Programming
- 2.2 Textbooks and Affiliated Software
- 2.3 Faculty vs. Departmental Control of the Curriculum
- 2.4 Teaching Process Simulators Through the Curriculum
- 2.5 Numerical and Analytical Approaches in Modeling of Physical Behavior
- 2.6 Laboratory Experiments Over the Internet

3.0 Process and Product Design
- 3.1 Molecular Modeling

4.0 Summary

Appendix A: Computing Practices in Industry: How Recent ChE Graduates Use Computing

Appendix B: Different ChE Departmental Approaches to Integration of Computing
- B.1 University of Texas
- B.2 Ben Gurion University
- B.3 University of Colorado
- B.4 University of Massachusetts
- B.5 University of Kentucky
- B.6 University of California, Santa Barbara

1.0 Computing and the Systems Approach in Chemical Engineering

The recent Frontiers in Chemical Engineering Education Workshops (see http://web.mit.edu/che-curriculum) identified the systems area as one of the three key focus areas for the future chemical engineering curriculum. As stated in the report of the workshops, "the systems component of the curriculum equips the graduate to:

- create and understand mathematical descriptions of physical phenomena,
- scale variables and perform order-of-magnitude analyses,
- structure and solve complex problems,
- manage large amounts of messy data, including missing data and information,
- resolve complex and sometimes contradictory issues of process design: sensitivity of solutions to assumptions, uncertainty in data, what-if questions, and process optimization."

The systems component of the curriculum trains students in computational tools for synthesis, analysis, design, and manipulation of chemical and biological processes. In the systems approach students learn how to convert scientific facts and principles of chemical and biological systems into engineering decisions. This includes computer-based methods for dynamic and steady-state simulation at multiple length and time scales, statistical analysis of data, sensitivity analysis, optimization, parameter estimation and system identification, design and analysis of feedback systems, methods for monitoring and diagnosis, and methods for design of products and processes.

2.0 Teaching of Computer Programming and Computer-based Courses

2.1 Teaching Computer Programming

In the area of computing software, there is a noticeable disconnect between industry and academia. Appendix A presents a survey of computing practices of recent graduates in chemical engineering. Typically Chemical Engineering Departments teach the use of MATLAB, MathCAD, Mathematica, or Maple but not the use of spreadsheets. Yet in industry, spreadsheet software (e.g., Excel) is the dominant computer package in use. Of course this may reflect the nature of many calculations that need to be performed by chemical engineers in industry, rather than a need to de-emphasize the teaching of sound numerical approaches in universities. There has been resistance to teaching spreadsheets, for example, it is difficult to analyze the logic in the code, but this appears to be changing. Another objection is that a spreadsheet approach can encourage the use of inaccurate or inefficient numerical calculations (no error control, etc.) For complex calculations, it may be better to program spreadsheets using Visual Basic, where programming logic is more transparent.

Many departments no longer require a course in a computer programming language such as Fortran, C, or C++. It has been suggested that teaching computer programming is analogous to teaching plane geometry. It is a way of thinking but you may not have to use it. On the other hand, without some programming ability, engineers are limited by the built-in capability of commercial software without any way to extend it.

Many chemical engineering departments are wrestling with the following questions:

- When should computing be introduced to the chemical engineering student?
- How should computer programming in chemical engineering be taught, and how much formal programming instruction on languages such as C should be provided (vs. usage of computing tools such as MATLAB, MathCad, spreadsheets, etc.)?
- Is a numerical methods course required and when does this occur in the course seqeunce? How many credit hours are needed?
- Should every course include some computing?

Since the mid-80's two approaches have been taken toward introductory computing for engineers. "CS 101" and the engineering tools approach. The CS 101 approach was catalyzed by the growth of computer science programs, which provided instruction in computer languages. The "CS 101" courses have migrated through several programming languages: Pascal, C, C++, and Java. In the engineering branch, software vehicles such as spreadsheets (first Lotus 123, then Quattro Pro, and now Excel), TK Solver, MathCad, and MATLAB have gradually pushed out programming languages (primarily Fortran). These languages are becoming endangered species in these courses.

The "CS 101" branch would claim a number of reasons for existence (Clough, 2003):

- engineers should learn fundamental concepts of programming and computer science
- computing should be taught by computer scientists, not engineers
- engineering faculty are not interested in teaching computing languages to their students
- these courses provide a significant number of student credit hours (SCH) and budgetary resources

The "Engineering Tools Approach" branch believes:

- engineering students need a solid grounding in problem solving with modern computing tools
- engineering students need the knowledge and tools required in their professions
- engineering computing and problem-solving are best taught by engineers
- these courses comprise a significant number of student credit hours (SCH) and budgetary resources

An in-depth assessment reveals that the two branches are complementary and that many/most engineering students can benefit from both courses. However, most curricula are too congested to make room for both.

When it comes to computing needs, faculty often confuse what is important for their students with what is important for themselves. Faculty needs, more often than not, align with their research interests and activities, and these may be disconnected from the needs of their undergraduate students. Also, faculty often have an incorrect impression of the computing needs of professionals by either being out of date or out of touch. Few discussions on computing needs proceed on the basis of evidence from alumni and employer surveys. Finally, computing is not part of the daily professional existence of most faculty and is not expected to be. Their computing skills can be oxidized, and most of their computing is carried out by their students.

The survey of industrial usage of computing in Appendix A indicates that a minority of recent graduates in chemical engineering actually perform programming on the job (although there is no clear definition of what constitutes programming in industry). The use of spreadsheets in chemical engineering practice is increasing. The application of spreadsheets in university courses are attractive because of student-driven usage. David Clough (Colorado) and Brice Carnahan (Michigan) have developed many examples of spreadsheet applications, as presented at the 2002 ASEE Chemical Engineering Summer School (see www.cache.org).

Textbooks and Affiliated Software

The fragmented nature of software tied to leading undergraduate textbooks makes integration difficult, e.g.,

(a) material and energy balances: Felder and Rousseau – EZ Solve; Himmelblau and Riggs – POLYMATH
(b) thermodynamics: Sandler – MathCAD; Kyle – POLYMATH; Elliott and Lira – various programs
(c) separations: Wankat – Aspen
(d) process control: Seborg, Edgar, Mellichamp – MATLAB; Marlin – MATLAB; Bequette – MATLAB; Riggs – MATLAB/Excel; Control Station is a standalone package used at a number of departments
(e) chemical reaction engineering: Fogler – POLYMATH, Rawlings and Ekerdt – Octave/MATLAB
(f) product and process design: Seider, Seader, Lewin – Aspen, HYSYS, CHEMCAD, PROII

See the CACHE report (www.cache.org), "Chemical Engineering Problems with Solutions in MATLAB, POLYMATH, Excel, Mathematica, Maple, and MathCad" for a comparison of these different software packages applied to solving a number of prototypical problems in different courses in chemical engineering.

In addition to these courses, many departments are teaching a statistics course, which involves still one more software package such as JMP, SAS, or Minitab. Clearly using a subset of these textbooks sequentially through the sophomore, junior, and senior years will require a student to learn up to five or more different software packages. Adding software packages outside of chemical engineering can push the total number of packages beyond ten. It would be desirable to keep the number of software packages below three or four if possible (note that Excel is not mentioned in (a)-(f) above although it is used with many of the textbooks listed). But usually textbooks are not chosen because of the bundled software. In addition, departments must address issues of software availability, licensing, cost, and providing software in computer labs vs. student-owned computers. A textbook that is closely coupled to a software package, a CD-ROM, or a website is clearly an attractive option.

2.3 Faculty Control vs. Departmental Control of the Curriculum

One view of academia is that the professor is a "high priest" in his/her course with a large amount of discretion to select course content as well as the textbook to be used. In some departments it is unclear if the Department chair or curriculum committees have the authority to influence the content of the core courses. It is reasonable to allow an individual faculty member to have some flexibility in coverage of certain topics in a given course. However, because of the tight coupling of prerequisite courses, there should be some way of guaranteeing that knowledge, ability, and skills (ABET KAS's) of students who complete a given course are predictable. Most of the core curricula should be predictable but some curricula do not need to be that prescriptive. Even when KAS's are covered, 20% of a given course can be left to the discretion of the instructor. Another issue is whether computing should be incorporated into a given chemical engineering course and be covered regardless of which faculty member teaches the course. Many departments are discussing this issue but based on a limited sample, it appears that most departments have not reached a consensus on coordinating content or incorporating computing through the curriculum.

2.4 Teaching Process Simulators Through the Curriculum

Several departments have found that the difficulty of integration of computing tools mentioned above can be avoided by more extensive use of process simulators. At Virginia Tech, ChE undergraduates have been using Aspen Plus and Aspen Dynamics to solve problems in all subjects. It is fairly straightforward to convert a steady-state model in Aspen Plus into a dynamic model (with PID control schemes) in Aspen Dynamics. The applicability of Aspen Plus to mass and energy balances, thermodynamics (physical and thermodynamic property analysis, estimation and regression), multicomponent separations, reactor design and process flowsheet simulation are well-known. In process control Aspen Dynamics enables the students to evaluate controller tuning, process dynamics, startup and shutdown, etc. HYSYS has similar features.

Recently, Version 2.0 of a CD-ROM, *Using Process Simulators in Chemical Engineering: A Multimedia Guide for the Core Curriculum* (Lewin et al., Wiley, 2003), has become available. Modules and tutorials are proved for self-paced instruction in the use of the process simulators to solve open-ended problems in courses on material and energy balances, thermodynamics, heat transfer, reactor design, separations, and product and process design. A 110-page document has been prepared for instructors suggesting the best instruction sequence and providing exercises and solutions, for each of the core courses (first introduced at the 2002 ASEE Chemical Engineering Summer School).

2.5 Numerical and Analytical Approaches in Modeling of Physical Behavior

Historically many engineering courses have previously been taught from an analytical viewpoint, but a transition is starting to occur, where numerical experiments are being gradually added. Problems and experiments should not be so simplified that they are not realistically formulated. Students are normally exposed to idealized fluid flow cases in the curriculum, for which application of theoretical concepts results in a solution of a one-dimensional ordinary differential equation or an algebraic equation. Therefore it is very easy for them to come away with the notion that theory is useless for most real-life situations.

Students should be able to select either analytical or numerical techniques to solve a problem, hence they should learn the advantages and disadvantages of either approach. Use of more sophisticated numerical tools such as CFD (computational fluid dynamics) will reduce the need to make many simplifying assumptions because you do not need as many assumptions to solve the problem numerically. Chemical Engineering students should understand that there are both numerical experiments and physical experiments. In some cases we can make observations from numerical experiments that you cannot see in physical data, but the converse is also true. This does not suggest that every experiment can be replaced with a simulator, but there should be a balance of the two approaches.

To prepare students for industrial practice, there should be a re-examination of the role of detailed analytical solutions. Is the purpose of some of these exercises the preparation of undergraduates for graduate school or industry? Today practicing engineers are not expected to carry out complex derivations in project work.

It is interesting to note that the public can run CFD simulators in science museums, therefore we ought to be able to teach this subject to chemical engineers. Once a fluid flow situation is analyzed theoretically or the governing principles are discussed, that same situation can be visualized using the computer. This visualization of the flow phenomena can significantly facilitate and enhance the learning process, especially for the visual learner. CFD software makes flow visualization easy. Students can simulate flow processes in a transient or steady state mode. Flow patterns can be displayed via velocity contours, velocity vector plots, or graphs of velocity profiles. A key element in flow visualization exercises is exploring the effect of different parameters. Using CFD, students can quickly change the size of the pipe, viscosity of the fluid, size of the particles, velocity of the feed, etc. and see the resulting changes in the flow behavior.

This type of parametric analysis also ties in nicely with a discussion of dimensionless groups and geometric and dynamic similarity.

While computing and visualization can increase understanding, educators do not want students to view such simulators as black boxes. In the fluid mechanics course, simulations can become a mathematical exercise with little intuition, unless the instructor has the students solve a simple problem by hand first. More work on the software tools is needed, and it is critical to match the software tool to the student's knowledge base.

Two specific recent packages which have been developed are FlowLab (a finite volume-based code) by Fluent, Inc. and FEMLAB (a finite element-based code) by Comsol, Inc. FlowLab allows students to solve fluid dynamics problems without requiring a long training period. Using carefully constructed examples, FlowLab allows students to get started immediately without having to spend the large time commitment to learn geometry and mesh creation skills required by traditional CFD software. Current exercises developed includes sudden expansion in a pipe, flow and heat transfer in a pipe, flow around a cylinder, and flow over a heated plate, among others. In addition, professors can create their own examples or customize the pre-defined ones.

FEMLAB provides ready-to-use application modes, where the user can build his/her own model by defining the relevant physical quantities rather than the equations directly. The software also allows for equation-based modeling, which gives the user the freedom to create his/her equations. FEMLAB's programming language is an extension of the MATLAB language; this feature gives much flexibility to the user. FEMLAB's graphical interface includes functions for automatic mesh generation of a user-defined geometry. Recently a k-e turbulence model has been added to its menu of options.

Seider et al. (2004) present the design of configured consumer products, which usually involves two or three-dimensional simulations. In Chapter 19, Product Design momentum and species balances in a 2-D plasma CVD reactor are employed to produce thin Si films using CFD packages such as FEMLAB. This illustrates where it is very effective for students to use CFD packages to optimize designs – even without understanding all of the physical and chemical interactions in the transport-reaction processes.

Even with these recent advances in educational CFD software, this computing technology has been slow to penetrate undergraduate transport and reactor engineering courses. A recent CACHE Survey of all chemical engineering departments in the U.S. on barriers to implementing CFD identified a lack of knowledge concerning available CFD resources, a lack of professor training in CFD, the ease of use and the long learning curve associated with using CFD software in a given course, and cost of CFD software.

2.6 Laboratory Experiments Over the Internet

Web-access of laboratory experiments enables real chemical engineering laboratory equipment to be controlled and monitored interactively by computers that are connected to the Internet, i.e., under the command of users over the Web. This capability is now available in the labs at U. Tennessee-Chattanooga as well as other schools such as U.Texas-Austin, Columbia University, University of Toledo and MIT. This would permit faculty and students from any university to run Web-connected experiments at any time of the day or night, any day of the week. The laboratory station computer operates the equipment (pumps, valves, heaters, relays, etc.), collects the data (pressure, temperature, position, speed, concentration, etc.) and sends it to the web user. The U. Tennessee site is accessed through the web address, http://chem.engr.utc.edu/, and even includes audio and video of the operating equipment.

All established chemical engineering programs are facing increased financial pressure to keep existing laboratory experiments up-to-date and in satisfactory operating condition. Major operating costs of unit operations laboratories include maintenance and teaching assistant support. Using highly automated experiments for remote operations will allow a drastic reduction in TA time requirements for those particular experiments. In addition, by sharing the operation of the experiments among several universities, there can be a pro rata reduction in maintenance costs. There is also the opportunity to add experimental assignments to a lecture class using this technology. In a lecture class, it may be desirable to have students individually or in small groups carry out an experiment, much like a homework assignment; in contrast, a traditional experiment would require continuous supervision by teaching assistants (e.g., one week of TA time for an entire class). Therefore using an internet-based experiment can greatly reduce the time commitment by the TA. However, traditional experiments will remain in the curriculum to give students "hands-on" exposure, but they can be augmented with internet labs.

3.0 Process and Product Design

Historically there has been a process design emphasis in the curriculum that is now transitioning to a dual product and process design emphasis. This means that a framework is needed to make process decisions in order to make structured products. This has added a performance layer, i.e., not just purity of the product. Given a structure, we can often predict at some level what the properties of the material are likely to be. The accuracy of the results and the methods used to treat them depend critically on the complexity of the structure as well as the availability of information on similar structures. For example, various quantitative structure property relationship (QSPR) models are available for the prediction of polymer properties. However, the inverse engineering design problem, designing structures given a set of desired properties is far more difficult. The market may demand or need a new material with a specific set of properties, yet given the properties it is extremely difficult to know which monomers to put together to make a polymer and what molecular weight the polymer should have. Today the inverse design problem is attacked empirically by the synthetic chemist with his/her wealth of knowledge based on intuition and on experience. A significant amount of work is already under way to develop the "holy grail" of materials design, namely, effective and powerful reverse-engineering software to solve the problem of going backwards from a set of desired properties to realistic chemical structures and material morphologies that may have these properties. After this is completed, a subsequent step would involve how to manufacture the desired new product.

A chapter on Molecular Structure Design in Seider et al. (2004) contains simple optimization procedures using GAMS to determine polymer repeat units, refrigrants, and solvents that have desired properties using group-contribution methods. Eventually, these will be replaced (and augmented) by molecular models.

Another subject related to product design is the scheduling of batch processes, which can be done using simple simulation techniques, as in BATCH PLUS and SUPERPRO DESIGNER. Hence design of optimal processing can be viewed as "product design" for specialty chemicals. Clearly, spreadsheets and optimization packages can also be used for many of these computations. Finally, the use of large databases and software systems for equipment sizing and purchase and installation cost estimation, such as ASPEN IPE, are being commonly used throughout the chemical industries for product and process design.

3.1 Molecular Modeling

A molecular-level understanding of chemical manufacturing processes would greatly aid the development of steady-state and dynamic models of these processes. Process modeling is extensively practiced by the chemical industry in order to optimize chemical processes. However, one needs to be able to develop a model of the process and then predict not only thermochemical and thermophysical properties but also accurate rate constants as input data for the process simulation. Another critical set of data needed for the models are thermophysical properties. These include such simple quantities as boiling points and also more complex phenomena such as vapor/liquid equilibria phase diagrams, diffusion, liquid densities, and the prediction of critical points. A key role of computational chemistry is to provide input parameters of increasing accuracy and reliability to the process simulations.

Under the NSF grant, "World Wide Web-Based Modules for Introduction of Molecular Simulation into the Chemical Engineering Curriculum", seven university experts in molecular simulations have worked with the CACHE Molecular Modeling Task Force (MMTF) to develop WWW-based modules to facilitate introduction of molecular simulation into the chemical engineering undergraduate curriculum. These teaching modules can be integrated directly into chemical engineering core undergraduate courses, supplying for the instructor and the student the appropriate linkage material between macroscopic concepts currently taught in these courses and molecular simulations designed to aid student understanding of the molecular underpinnings of the phenomena. Modules are centered around Java Applets that run the molecular simulations and provide an "experimental" simulation platform for students to explore concepts. In addition, modules contain instructor materials, fundamental tutorials, student problems, and assessment materials.

MMTF has designed a consistent web-based interface that organizes all of the material in each module and developed scripts using *perl* that ease the job of putting the written material into this common format. The developer of a module must construct simple text files, perhaps with HTML markup that permits inclusion of figures and tables. Then he or she runs the files through the *perl* script, which adds HTML formatting and links to put the set of files into the common configuration. The files are uploaded to the module site for anyone to access. This site is perhaps best accessed through the *Etomica site*. *Etomica* is a Java-based support environment developed for the modules project, which has now been expanded for other applications: go to http://www.ccr.buffalo.edu/eto and click on the "modules" link in the navigation bar on the left.

Following is a list of phenomena and concepts for which modules are completed or planned:

- Chemical reaction equilibrium
- Osmosis
- Diffusion
- Molecular dynamics
- Normal modes of a solid
- Chemical reaction kinetics
- Dissipative particle dynamics
- Surface tension
- Crystal viewer
- Joule-Thomson expansion
- Self assembly
- Chemical potential
- Multicomponent phase equilibrium
- Heat transfer
- Atomic billiards
- Viscosity

Contact David Kofke at the University of Buffalo for further information.

4.0 Conclusions and Recommendations

One way to foster renewal of the curriculum is to identify departments where curriculum revision is being carried out and to evaluate best computing practices and current trends. There may not be one answer because of different constraints various universities operate under, such as number of faculty in the department and whether computing courses are taught outside the department. Appendix B presents a number of different departmental approaches to integrating computing through the curriculum.

CACHE makes the following recommendations on computing through the curriculum:

(1) The "systems" component of chemical engineering is tied closely to computing tools, so strengthening the systems area as recommended by the Frontiers of Chemical Engineering Workshops should also strengthen the use of computing as well.

(2) There is increasing pressure on the total number of hours in the curriculum, especially with the addition of life science courses. Departments should continue to re-examine whether a formal three or four credit hour computer programming course is required for the chemical engineering degree (vs. teaching how to use software or write m-files in MATLAB, for example). The chemical engineering computing course also provides students with a valuable experience in quantitative problem-solving.

(3) The number of software tools that implement numerical methods used by students should be minimized; departmental agreement on software used in each course should be reached within the faculty. Faculty need to reach consensus on how student computing skills can grow systematically through evaluating each course in the curriculum.

(4) Courses such as transport phenomena and thermodynamics offer new possibilities for introduction of computing physical and chemical behavior, such as with computational fluid dynamics or molecular modeling. Process design can add a product design emphasis by using such tools as well.

(5) Internet-based laboratories offer a new means of bringing live experimental data into lecture-based courses, in order to reinforce theoretical concepts.

(6) To prepare students to optimize process designs, it helps to expose students to process simulators for solution of a problem(s) in the core courses of the chemical engineering curriculum. Also, as software develops and product design is added at the senior level, instructors must select from among optimization packages (such as GAMS), batch process simulators (such as BATCH PLUS and SUPERPRO DESIGNER), and packages for estimating equipment sizes and installation costs (such as Aspen IPE). The use of comprehensive software packages and databases is common in industrial design and needs to be introduced in design courses and utilized for solution of design projects.

Acknowledgments

Contributors to this paper include T.F. Edgar, W.D. Seider, D.E. Clough, J. Curtis, B.L. Knutson, P.R. Westmoreland, J.J. Siirola, Chau-Chyun Chen, G.V. Reklaitis, R. LaRoche, J.B. Rawlings, E.M. Rosen, D. Kofke, M. Cutlip, and M. Shacham.

References

Clough, D.E., “ChE’s Teaching Introductory Computing to ChE Students – A Modern Computing Course with Emphasis on Problem Solving and Programming”, *ASEE Annual Meeting*, 2002.

Lewin, D.R. et al., “Using Process Simulators in Chemical Engineering”, CD-Rom for Seider et al. (2004) textbook.

Seider, W.D., Seader, J.D., and Lewin, D.R., *Product and Process Design Principles,* Wiley, New York, 2004.

Appendix A: Computing Practices in Industry: How Recent ChE Graduates Use Computing

– Powerpoint Presentation

Appendix B: Different ChE Departmental Approaches to Integration of Computing

B.1. University of Texas-Austin - Integration of Computing

During 2001-2002 The Department of Chemical Engineering at UT-Austin revamped the computing thread in its curriculum in order to strengthen student background in computing. This action was in response to student and faculty dissatisfaction with the depth and continuity of computer training over the four years of the program. The curriculum modifications included:

1. Adding a new ChE freshman course, ChE 210: Introduction to Computing, focusing on basics of computing, MATLAB, and Excel.

2. Changing ChE 448 to ChE 348 to focus on Numerical Methods in Chemical Engineering and Problem Solving, in the second semester – sophomore year.

3. Changing the existing junior lab course (3 credit hours to two two-credit hour courses: ChE253K: Introduction to Statistics and Data Analysis, and ChE 253M, Fundamental Measurements Laboratory.

In addition, reinforcement of the computing tools was implemented in core ChE courses by identifying prototype problems where these tools (mostly Excel and MATLAB) would be used. Table B.1 shows the layout of the required courses and where computing is employed.

Table B.1. Computing Tools Used in the ChE Curricula at UT-Austin

Year	Computing Activities
Freshman	Introduction to MATLAB, Excel (second semester)
Sophomore	Material and Energy Balances (Excel) Numerical Methods (MATLAB, Excel) Transport Phenomena (Excel)
Junior	Thermodynamics (Excel) Fluid Flow/Heat Transfer (MATLAB, CFD) Statistics (JMP, Excel) Separations (Aspen, Excel) Measurements Laboratory (JMP, Excel)
Senior	Reactor Design (MATLAB, Excel) Process Control (MATLAB, Excel) Unit Operations Laboratory (data acquisition/control, Excel) Process Design (Aspen, @Risk, Excel)

The advantage of the proposed changes is that computing and numerical analysis are now spread uniformly over the first three years of the program, namely two hours on computer software tools in the second semester of the freshman year (ChE 210), three hours of numerical analysis in the second semester of the sophomore year (ChE 348), and two hours of statistics in the third year, increasing statistics instruction from one hour to two hours (ChE 253K). The doubling of coverage of statistics was in response to feedback from industry on the importance of statistics in chemical engineering practice. Note that even with all of these changes, there was no net increase in the number of credit hours for the degree.

The objectives of the two modified computing courses (210, 348) are as follows. Upon completion of ChE 210 students should:

- Understand basic computer architecture and internal number representation.
- Have an appreciation for limitations in numerical accuracy.
- Be able to construct plots, fit data, and build new functions using Microsoft Excel.
- Demonstrate ability to create complex programs in a programming environment such as MATLAB.

Upon completion of ChE 348 students should:

- Be able to identify and formulate methods to solve specific classes of numerical problems, including linear equations, nonlinear equations, numerical integration (quadrature), least-squares curve-fitting, minimization of functions, and differential equations.
- Understand how software can be used to solve each class of problem.
- Know limitations of each method.

Detailed outlines of the three new/revised courses (210, 348, 253K) are given below. For each topic, the number of one-hour lectures are given in parentheses.

ChE 210 (Introduction to Computing)

1. Introduction to Computers (5)
    - History of computing devices
    - Modern computer architecture
    - Number representation and round-off
    - Internet, web, HTML

2. Spreadsheets (7)
   - Simple cell arithmetic
   - Plotting data – data visualization – good graphics
   - Solver
   - Visual Basic for applications

3. Programming Concepts (9)
   - Problem analysis and specification
   - Algorithms and control structures
   - Flow Charts and pseudocode
   - Sequential processing (order of precedence, arithmetic operations)
   - Selection structures (if-end, if-else-end, if-else if-else-end)
   - Repetition structures (for, while)
   - Comparison operators and Boolean expressions

4. MATLAB Programming (8)
   - Matrices and vectors
   - Plotting
   - Scripts
   - Functions
   - Selection, repetition and logicals

ChE 348 (Numerical Analysis)

1. Review of Program Organization and Structure (4)
   - Overview of course and MATLAB
   - Review of programming, control structures
   - Review of Taylor series
   - Errors, accuracy and stability

2. Matrices (3)
   - Elementary matrix-vector operations
   - Properties of matrix operations: eigenvalues, diagonalization
   - MATLAB operations

3. Linear equations (3)
   - Gaussian elimination/partial pivoting
   - Tridiagonal and band diagonal matrices

4. Single Nonlinear Equations (3)
   - Graphical solution
   - Newton, secant, Broyden methods

5. Multiple Nonlinear Equations (5)
   - Graphing zero contours

Newton's method, partial derivatives
MATLAB: fsolve
Example: Multiple reactions, CSTR

6. Differential Equations (7)
   Review of ODE's: linear vs. nonlinear ODE's, order of ODE's, linear first order ODE's (integrating factors), and solutions of second order ODE's.
   Quadrature: Simpson, Trapezoidal methods
   Numerical integration of initial value problems, Runge Kutta method
   Shooting methods

7. Multiple ODE's (3)
   Simplest method: solving reaction network problems with multiple reactions
   Connections to multiple algebraic equations (eigenvalues etc.)

8. PDE's (3)
   Parabolic PDE's (heat conduction problem)
   Other boundary conditions

9. Optimization (3)

As time permits, one or more of the following topics: Monte Carlo integration, molecular dynamics as an example of second order ODE's, stability and chaos

ChE 253K (Introduction to Statistics and Data Analysis)

1. Introduction (2)
   Discrete vs. continuous
   Variance of measurements
   Value of statistical analysis

2. Descriptive Statistics (3)
   Data sorting
   Frequency tables
   Stem and leaf plots
   Histograms
   Pareto plots
   Ogive plots

3. Probability (2)
   Defining probability
   Counting techniques, permutation and combination
   Additivity and Multiplicative rules
   Bayes' rule

4. Working with discrete random variables and probability distributions (2)
    - Define discrete random variables and continuous variables
    - Binomial distribution
    - Hypergeometric distribution
    - Poisson distribution

5. Working with Continuous Probability Distributions (2)
    - Normal distribution
    - Normal approximation to the binomial
    - Chi-Squared distribution

6. Functions of Random Variables (1)
    - Moments and moment generating functions

7. One and two Sample Estimations (3)
    - Statistical inference
    - Estimating the mean
    - Standard error
    - Tolerance limits
    - Estimating the difference between two means
    - Paired observations
    - Estimating variance
    - Estimating the ratio of variances

8. Hypothesis Testing (3)
    - Concepts
    - One and two tailed tests
    - Use of p-values
    - Choice of sample size
    - Tests of means
    - Tests of Variance

9. Linear Regression and Correlation (4)
    - Least square estimators
    - Analysis of variance approach
    - Linear regression case studies

10. Second Factorial Experiments (3)
    - Concepts of statistical experimental design and response surface analysis
    - Introduction to JMP for Box Behnken, etc.
    - Design of experiments and Response surface analysis with JMP

11. Statistical Process Control (2)

  - Nature of control limits
  - Purposes of control charts
  - X-bar charts
  - R-bar charts
  - Cusum control charts

B.2 Ben-Gurion University

Most students find it difficult to learn programming and in practice very few of them do any programming after finishing the basic course on the subject. After over 35 years that programming has been taught in the Chemical Engineering Departments, many schools now consider dropping those studies altogether because of their ineffectiveness.

At the Ben-Gurion University we have removed the required programming course from the curriculum two years ago and at the same time we extended an existing “Introduction to Personal Computers (IPC)” course by incorporating MATLAB programming into it. The main objective of the IPC course is to enable the students to solve engineering problems that involve complex consecutive calculations, solving linear and nonlinear algebraic and ordinary differential equations, and carrying out multiple linear and polynomial regression, using POLYMATH, Excel and MATLAB. To enhance the learning effectiveness the following advanced features are included in the course:

Based on real life problems – the study of each subject starts by presenting a real life problem, which requires numerical solution using one of the techniques mentioned above. Typical benchmark problems used include the calculation of thermodynamic properties using a cubic equation of state (Shacham et al, 2003), calculation of flow rate in draining a tank (Brauner and Shacham, 1994), calculation of adiabatic flame temperature (Shacham, 1998) and correlation of vapor pressure data using the Clapeyron, Antoine and Riedel’s equations (Shacham et al, 1996).

Multi-stage problems – the problems require computer solution in several stages. Preliminary stages are solvable with the “user friendly” software packages (POLYMATH, Excel) while advanced stages require MATLAB programming (Shacham et al, 2003).

Programming by modification – solved examples are provided. The students are encourages to modify and extend the solved examples to solve their assignments.

Self – grading – students check and grade their own homework assignments and exams. The self-test program provides feedback on the errors made (Shacham, 1998). Most students will not be satisfied unless they get a working program that yields correct results.

The problem-based approach ensures that the students understand the need in learning programming and this increases their motivation. Using *multi-state problems* and *programming by modification* ensures that the students can get a working program in a reasonably length of time without losing faith in their programming abilities. The *self-grading* aspect provides an additional incentive to get a working program that yields correct results.

The final exam of the course in this last semester included the use of the “secant” method for solution of the adiabatic flame temperature problem (Shacham, 1998) with MATLAB programming. The students had an hour and a half to complete this assignment and 75% of them (40 students) managed to get a working program that yielded the correct solution. This clearly demonstrates the effectiveness of the teaching techniques that have been used in the IPC course.

Introduction to Modeling and Computation for Chemical Engineers with POLYMATH, Excel and MATLAB

Chapter 1: Complex Consecutive Calculations (pdf file, 233 KB)
Homework assignment **1:** Calculation of Molar Volume and Compressibility
Factor of a Gas Using the Redlich – Kwong Equation of State.
a. Problem Definition
b. Self Test

Chapter 2: Iterative Solution of a Nonlinear Equation (pdf file, 256 KB)
Homework assignment **2:** Calculation of Adiabatic Flame Temperature
a. Problem Definition
b. Self Test

Homework assignment **3:** Calculation of Molar Volume and Compressibility
Factor Using Various Equations of State.
a. Problem Definition
b. Self Test

Homework assignment **4:** Bubble Point and Dew Point for an Ideal Multi-component Mixture
a. Problem Definition
b. Self Test

Chapter 3: Matrix Operations and Solution of Systems of Linear Equations (pdf file, 193 KB)

Chapter 4: Multiple-Linear, Polynomial and Nonlinear Regression. Part I (pdf file, 215 KB), Part II (pdf file, 310 KB)
Homework assignment **5:** Fitting Correlations (Regression Models) to Vapor Pressure Data
a. Problem Definition
b. Self Test

Homework assignment **6:** Linearization of the Antoine Equation
a. Problem Definition
b. Self Test

Chapter 5. Introduction to Solution of Systems of Nonlinear Algebraic Equations

Chapter 6. Introduction to solution of Systems of Ordinary Differential Equations (pdf file, 263 KB)

Comments, Corrections and Suggestions

Comments, corrections and suggestions regarding this course are welcome and should be sent to: shacham@bgumail.bgu.ac.il

References (Section B.2)

Brauner, N. and M. Shacham, “Numerical Experiments in Fluid Mechanics with a Tank and Draining Pipe”, *Comput. Appl. Eng. Educ*, **2**(3), 175-183 (1994).

Shacham, M., N. Brauner and M.B. Cutlip, “Replacing the Graph Paper by Interactive Software in Modeling and Analysis of Experimental Data”, *Comput. Appl. Eng. Educ.*, **4**(3), 241-251 (1996).

Shacham, M., “Computer Based Exams in Undergraduate Engineering Courses”, *Comput. Appl. Eng. Educ*., **6**(3) 3, 210-209 (1998).

Shacham, M., N. Brauner and M.B. Cutlip, “An Exercise for Practicing Programming in the ChE Curriculum”, *Chem. Eng. Educ.*, **37**(2), 148 (2003).

Appendix B.3 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 Matchcad 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.

B.4. University of Kentucky

The incorporation of computing through the curriculum is being addressed at the University of Kentucky with a goal of achieving the following objectives:

- Student proficiency in computing skills representative of the breadth of computing tools encountered in industrial practice.
- Demonstration of student's ability to for in-depth analysis using selected individual computing tools.
- Demonstration of student confidence in their ability to employ computing tools and to learn new tools
- Cohesive and efficient approach to introducing and using computing tools through the curriculum.

These objectives are being achieved through changes in the delivery of current courses as well as proposed curriculum reform. Through surveys of current students, alumni, co-op employers, and our faculty, the department clearly identified deficiencies in our previous approach to introducing and using computing tools. Namely, our required programming language course (Fortran or C++), which is taught outside our department, was not preparing students for engineering analysis. In addition, the programming language was not being sufficiently utilized and reinforced in the subsequent core chemical engineering courses for which it was a prerequisite. Numerical methods packages were being utilized in core chemical engineering courses, but the use of multiple software packages (e.g, Maple, MATLAB and Mathematica) limited the students' perceived proficiency in the use of computing tools. Further, students were not well-prepared to use chemical engineering simulation packages (e.g., Aspen or Chemcad) at the level required in senior-level design projects. Finally, knowledge of Excel was required for a broad range of core classes but was not introduced in a systematic manner.

These survey results led the faculty to carefully evaluate the choice of computing tools and the implementation of these tools in our curriculum. The following steps have been undertaken to incorporate computing tools through our curriculum:

1. A clear description of the objectives relating to computing in our curriculum was developed (as stated above).
2. A limited number of computing tools were selected and a plan for their introduction and implementation in the curriculum was formulated.
3. The relevance of a programming language in our curriculum was formally assessed.
4. The development of a new computing tools course within our chemical engineering curriculum was examined.

As a department, we have chosen to focus on three computing tools (a numerical methods package, a chemical engineering simulation package, and a spreadsheet program) and the implementation of these tools through the curriculum (Table B.1). Maple was chosen over other commercial numerical packages such as MATLAB because it is widely available, sufficient for most undergraduate problems, and instills the basis for learning more powerful numerical packages. In addition to being used by our chemical engineering faculty, Maple is also introduced in the freshman and sophomore-level calculus courses at the University of Kentucky. Aspen has been our primary unit operations simulation package, and continues to be supported in recent textbooks examples. Computer-based tutorials (D.R. Lewin, W.D. Seider, J.D. Seader, E. Dassau, J. Golbert, D. Goldberg, M.J. Fucci, and R.B. Nathanson, "Using Process Simulators in Chemical Engineering: A Multimedia Guide for the Core Curriculum" Distributed by John Wiley & Sons, Inc,. Version 2) has greatly enhanced the ability to incorporate unit operations simulation packages into our core courses. Recognizing its importance in industrial practice and its practical value, Excel is now emphasized as a primary computing tool in our curriculum. Training in Excel in our unit operations laboratory is coupled with statistical analysis, thereby addressing two educational needs of our program. The overall goal of this implementation is to provide students with fewer computing tools, but to increase their proficiency and confidence in these tools by reinforcing their use throughout the curriculum. The exception to our "preferred list" of computing tools occurs in Process Control, where Simulink, a MATLAB tool, is a standard tool for the simulation and analysis of control systems.

Table B.1. Current implementation of computing tools at the University of Kentucky

Year	**1st Semester**	**2nd Semester**
Freshman	Calculus I: Maple	Calculus II: Maple CS Programming Language
Sophomore	Calculus III: Maple CME Process Principles: ASPEN	Calculus IV: Maple CME Thermodynamics: ASPEN
Junior	CME Separation Processes: ASPEN and Excel	CME Process Modeling: Maple
Senior	CME Unit Ops. Lab: Excel CME Reactor Design: ASPEN CME Process Design I: ASPEN and Excel	CME Process Control: Simulink CME Process Design II: ASPEN and Excel

Our current plan of implementation of computing tools reflects our department's choice to move away from a formal programming language course. Thus, we are piloting a 2 credit hour CME course (CME 1xx *Computational Tools for Chemical Engineers*, Spring 2004) aimed at freshmen-level students. Our purpose for the course is the meaningful introduction of computing tools (Excel, Maple, and ASPEN) which will be used throughout the curriculum. One goal of the new course is proficiency in Excel, including mastery of the basic interface, graphing, regression, and optimization. Knowledge of the tools of Maple is the goal for the second section of the course. We will focus on simple methods used to numerically solve problems (symbolic math with Maple is covered in our Calculus sequence) and avoid a "black box" treatment. In this freshman-level course, only a basic introduction of ASPEN for the solution of simple problems is an appropriate goal. In addition to developing proficiency with the user interfaces of these packages, the focus will be on numerical methods (Newton-Raphson, solving linear equation sets, numerical differentiation and integration) used across all packages. This course will incorporate *chemical engineering* problems that the students will see again in future core courses (e.g., flash calculations). *Computational Tools for Chemical Engineers* will be offered as an optional course during its pilot period, to be substituted for our 2 hour programming language course. As we move toward the full implementation of the freshman-level Computational Tools course as a required course, we anticipate that our integration of computing tools in our curriculum will be described by Table B.2. The placement of this course in the freshman year will allow us to increase our reliance on computing tools, particularly Excel, in the sophomore year.

Table B.2. Proposed implementation of computing tools at the University of Kentucky

Year	**1st Semester**	**2nd Semester**
Freshman	Calculus I: Maple	Calculus II: Maple CME Computational Tools: Maple, ASPEN, and Excel
Sophomore	Calculus III: Maple CME Process Principles: ASPEN and Excel	Calculus IV: Maple CME Thermodynamics: ASPEN and Excel
Junior	CME Separation Processes: ASPEN and Excel	CME Process Modeling: Maple
Senior	CME Unit Ops. Lab: Excel CME Reactor Design: ASPEN CME Process Design I: ASPEN and Excel	CME Process Control: Simulink CME Process Design II: ASPEN and Excel

B.5. University of Massachusetts

see pdf file