AC 2007-1949: VERTICAL INTEGRATION OF MATLAB ACROSS ENGINEERING CURRICULA: SYSTEMIC CURRICULAR CHANGE BY SMALL STEPS

Jon Sticklen, Michigan State University

Daina Briedis, Michigan State University

Mark Urban-Lurain, Michigan State University

Timothy Hinds, Michigan State University

VERTICAL INTEGRATION OF MATLAB ACROSS ENGINEERING CURRICULA: SYSTEMATIC CURRICULAR CHANGE BY SMALL STEPS

Introduction

In the engineering workplace, newly minted graduates from our engineering programs are expected to be facile in formulating well-defined problems, and in selecting an appropriate tool with which to develop a solution once a problem is well posed. Typically in the engineering academic community, we have considered the analytical tools of traditional mathematics as the primary solution tools our students must systematically master. The language of mathematics is indeed indispensable to us for representing a well-posed engineering problem. But solution tools bundled into the modern computational environments provide a second and complementary solution capability, and one that is core for problems with no closed-form solution.

Undergraduates in the engineering disciplines are certainly *exposed* to solution tools that are computer based. In general however, the *systematic use* of modern computer environments does not currently receive the formal attention in our curricula that traditional mathematics receives. Yet we expect our students to solve problems as assigned using such environments. This puts engineering students in a bind. Without systematic and repeated use of a given computer environment over the entire undergraduate student experience, when faculty tell students to solve a rich problem using a computer tool, the students spend inordinate time (re)learning the computer tool, and *not* focusing on the concept which the faculty member had in mind. We should not be surprised when a senior-level undergraduate comments "I don't know any MATLAB® - how can I solve this problem?" when the senior had a freshman course in MATLAB two years prior, yet had not used MATLAB in the sophomore or junior years. Two years of non-use of any tool will lead to forgetting key details for almost anyone.

In the College of Engineering of Michigan State University, the computer environment of choice for problem solving at the undergraduate level is MATLAB. This selection was made a number of years ago by a college-level ad hoc committee. The choice of MATLAB is well reasoned in our belief, both for intrinsic attributes of the environment, but also and increasingly because MATLAB is becoming a de facto standard in many university engineering curricula reflecting the increasing use of MATLAB in industry. In what follows we use MATLAB as our point of reference, but it is important to note here that our arguments apply with equal force should a given university select Mathematica, or MathCad, or … Systematic and sustained use of the computational environment is central, no matter what the specific selection might be.

Currently, the authors of this paper are engaged in an interdisciplinary collaboration to foster and promote course-level integration of MATLAB into most undergraduate engineering majors, with a long-term goal of *curriculum-level integration*. Our efforts have included faculty quick-start seminars in MATLAB basics, providing assistance to faculty who are developing homework assignments that use MATLAB, beginning the development of a library of web-available "How To" screen movies showing specific steps for performing given operations in MATLAB, and - at

the curricular level - developing course targets we will use as key islands in which MATLAB will be introduced into the curriculum, and from which we will expand.

The purpose of the current paper is to play the role of "think piece" or catalyst. A companion paper in ASEE 07 in the Chemical Engineering program (lead author, Prof. D. Briedis) will describe the quantitative results we have seen to date in our major prototypical area: Chemical Engineering.

There are two simultaneous threads in the current paper. The first is about the importance of integrating computational tool use through our engineering curricula, and about how we have approached that task. The second is that our work provides an exemplar of systematic curricular reform, and as such the lessons we have learned may be of broad general interest.

In the current report we will describe in detail the background that led us to the belief that curricular change is necessary towards *systematic* use of MATLAB in our programs. We go on to give background on the broad area of curricular reform as relates to our needs-driven situation. We then describe the specific initial projects we have undertaken on our path to reach the broad goal of MATLAB infusion across our undergraduate engineering programs. We end with a summary and a statement of our future plans.

Nature of the Problem We Address: Computational Tool Use in Engineering Curricula

Undergraduate education in engineering has been generally successful over the last fifty years as determined by the most important metric – a well-educated and productive cadre of effective engineers. However, critics have rightly pointed out increasing problems in the nation's engineering curricula and resultant general shortcomings of engineering graduates.

Among the broad scale problems there are not enough students pursuing and graduating from engineering programs, especially when viewed on a national need basis. A loss of students early in the engineering curriculum is part of the difficulty, as is a proportionately higher loss of females and minorities. These losses result in a deficit in numbers of *role models* for these groups, and a consequent longer-term feedback situation yielding lower and lower numbers of females and minorities in the engineering and technology professions.

One of the most pervasive impediments for students deciding against engineering lies in the student perception that our curricula are not integrated programs. A specific and important example is the perceived disconnect between computational tools for technical problem solving and the technical engineering disciplines the students hope to pursue. This impediment is apparent in student perceptions about first year courses that focus on computational environments like MATLAB.

The shortcomings pointed out by critics of engineering undergraduate programs take many additional faces. But many are traceable to shortcomings in the core-defining characteristic of an effective engineer: strong problem solving ability and joy in problem solving. Effective problem solving is predicated on (a) thorough understanding of technical background material required

for the problem at hand or an ability to obtain that understanding, (b) ability to integrate background material, (c) ability to sharpen a stated problem and produce a well-structured problem from an ill-structured problem, (d) ability to apply the background material systematically and effectively to the problem, (e) ability to critically interpret the results of the problem solving, and (f) ability to communicate the results of the problem solving. Underlying and pervasive through the preceding enumeration is the ability to work as part of a team towards the problem-solving goal.

Engineering curricula have focused strongly on criterion (a) above to the detriment of the other items in the list. Indeed, many if not most engineering classes have focused on the first aspect of (a): a through grounding in the "basics" of a given discipline as "delivered" through lecture. A slow but steady evolution of "content tyranny" to more and more reliance on "lecturing about" more and more material is a reflection of exploding amounts of knowledge in the engineering disciplines. Yet with ever more "knowledge to be imparted," engineering students find themselves with so many details to master that they often lose sight of the goal: effective problem solving predicated on *integrated* understanding of technical material.

A specific example of this general problem lies in the integrated student understanding of computer-based computational tools that support technical problem solving. General computational environments such as MATLAB, MathCad, and Mathematica are all versatile and capable. But because of the varied background training of engineering faculty, most engineering faculty have individually learned one or another of these computational environments. Typical undergraduate engineering curricula thus reflect professor preference in the designation of specific computational tools to be used to complete student problem sets and projects. The implicit assumption has been that undergraduate engineering students can simply "pick up" a specified computational tool and apply it to assigned problems. This assumption is false.

Modern computational environments are replete with many "features" that can each be leveraged for a given class of problem. However, this "high power" comes at a high price: a steep learning curve for students. A typical engineering undergraduate has a difficult time in applying the tools in a computational environment like MATLAB. Failure to remember, for example, that in MATLAB one routinely must build data vectors in order to utilize the rich numerical plotting capability of MATLAB can block student use of MATLAB for visualizing data in plotted form. Failure to remember (or simply not knowing) how MATLAB in general deals with vector data structures can thwart the entire problem solving process, or at least the effective use of the MATLAB tool in that process.

The confluence of the versatility and complexity of modern computing environments with the propensity of faculty to select their "favorite tool" for assigned use to students in completing problem sets and projects leaves undergraduate students in a largely untenable position. They are told to use a complex environment that either they have not touched before, or one that they have not used for several years. The result is that students seek and use cookbook approaches to the particular assigned task without learning integrating principles for use of the target computational environment. Cookbook use precludes student growth in understanding how the tool may be used to advantage in a different problem setting.

One might suppose that once a student takes a freshman-level course emphasizing (e.g.) MATLAB, that the student would understand the basic principles that underlay any computational environment. Then the "transfer" of understanding from having learned a first computational environment (e.g., MATLAB) could be made to a new computational environment (e.g., Mathematica) with little student effort. In practice, this does not occur, and the learning literature clarifies why. In order to support transference of experience from one context to another, a student must thoroughly understand and integrate knowledge at a conceptual level from the first context. In order to master even a working subset of MATLAB, more than several "exposures" are necessary.

For undergraduate students to be competent with MATLAB, repeated use in different contexts is central. An analogous exemplar suggests that direct transfer of expertise from one setting to another is not to be generally expected.

Consider the way that students learned to use and apply slide rules in the past. Before powerful handheld calculators or computer-based tools were widely available to engineering students, they relied on slide rules for doing assigned problems.[1] Students *mastered* slide rule use in those former times for two reasons. First, slide rules were in fact much simpler than tools like MATLAB. Second, and more importantly, students used slide rules throughout their entire engineering careers in college and on into their professional careers. Repeated use of computational tools leads to familiarity, and to what now is called "pervasive computing" in the sense that the slide rule was simply a well-known tool to the student. Most engineering undergraduate students do not develop like familiarity with computational environments like MATLAB for the simple reason they do not achieve self-reliance with MATLAB, largely because they do not use it throughout their undergraduate experience.

The crux issue is that higher order learning, i.e. concept oriented learning, is necessary before transference across problem settings is effective.[2] To enable higher order learning in the context of a computer toolset, the tools must be mastered, and *re*-mastered thus enabling the learner to be confident of his detailed tool use while he focuses on content issues. Learning to effectively problem solve with computer tools is a two dimensional situation: the computer tool itself is one dimension and the conceptual content of the problem being attacked is the other. As our students work their way thorough our curricula, they should increasingly be focused on the content issues of their disciplines, and need to "worry" about the computer tool they are using should be important only to the extent of expanding their knowledge of that tool.

The use of MATLAB through the entire undergraduate experience can be conceptualized as a vertical slice through an engineering curriculum, and with the goal of developing MATLAB self-sufficiency via repeated use. Our long-term vision is that this vertical slice will touch all individual engineering courses that a student would take. Our narrower initial goal is to develop and implement a partial MATLAB vertical slice through the Chemical Engineering and Materials Science Department (CHEMS) at Michigan State University (MSU).

A key question is how we are to achieve the necessary faculty consensus to enable the curricular reform that results in integrated MATLAB use through the curriculum. We believe the answer lies in a standard tenant of good engineering: when faced with a complex task, the task should be

decomposed into manageable subtasks. Instead of developing a blueprint for total change followed by implementation of the entire blueprint across the curriculum, we believe that faculty consensus can be constructed *piecemeal* by building, from the bottom-up, pairwise linkages between courses such that each linkage will reinforce and build on prior student experience with MATLAB. We envision pairwise linkages of two types. A soft link is one in which conceptual uses of MATLAB in an earlier course is assumed and built upon in the higher-level course of the linkage. A hard link is formed by, for example, term project teams consist of students from both courses.

It is our working belief that by implementing over time a covering set of pairwise linkages across a curriculum that faculty support will evolve towards more support for additional linkages and, more importantly, towards their continued maintenance. Like a balance beam, the scales of faculty support would eventually tip to strongly favor the integrated curriculum based on the set of course linkage pairs.

Relevant Background on Curricular Change

Although our proposed project focuses on only one element of undergraduate engineering curricula only, use of computational tools, our project even at the Phase 1 level addresses issues beyond one course, and thus deals with issues of change across the curriculum. It is instructive to briefly recount recent history of efforts aimed at systemic curricular change in engineering programs.

In 1991, the National Research Council[3] criticized undergraduate engineering curricula for not reflecting the shifting needs of the engineering profession by saying that these curricula are "lacking the essential interdisciplinary character of modern design practice" (p. 4). As a result, NRC claimed, engineering graduates are poorly prepared to utilize "scientific, mathematical, and analytical knowledge in the design of high-quality components, processes, and systems". The ABET Engineering Criteria (earlier called Engineering Criteria 2000) reinforce these perspectives as has the National Science Foundation in the last decade.[4]

Curricular reform efforts have focused on developing new paradigms for engineering education, including an emphasis on active student learning and application of knowledge (including using design) rather than passive data gathering, faculty acting as mentors and facilitators rather than as lecturers, integration of disciplinary knowledge instead of emphasis on isolated facts, emphasis on deep problem solving including problem specification instead of "plug and chug" application of equations, innovative forms of student assessment focusing on improvement, and a variety of non-technical skills such as communication and teamwork central to the workplace.[5] Bordogna et al.[6] argues that more holistic curricula are needed that weave process knowledge (such as much understanding associated with the use of computational tools) and fact-based knowledge throughout the undergraduate experience. In spite of effective projects funded by NSF, its partner agencies, industry and postsecondary institutions, challenges remain in creating and institutionalizing reform initiatives to enhance learning outcomes in science, technology, engineering, and mathematics (STEM) fields.

There are several reasons for this apparent lack of adaptation. An experience to attempt change at MIT is instructive. The MIT Department of Aeronautics and Astronautics incorporated active learning strategies and assessment tools into their Unified Engineering course after a two-year strategic planning process that involved all faculty in the department.[7] As they discovered, "changing how we teach is more difficult than changing what we teach." (p. T2A-15) This change required not only faculty buy-in, but also administrative and institutional support. There is a two-fold message here. The first is that change in content or pedagogical delivery methods is, in fact, difficult for faculty. Second, systemic change, particularly if attempted in a "revolutionary" way (with all change to be implemented simultaneously), is yet more difficult.

Although the several NSF coalition program goals have in some sense set a standard for curricular program change (such as those stipulated through the Engineering Coalition of Schools for Excellence in Education and Leadership, ECSEL), such change is often difficult to accomplish on a typical campus because of the necessary faculty buy-in.[8] Put more strongly, the usually encountered bottleneck constraining systemic curricular change is achievement of the necessary level of faculty buy-in. The approach followed in ECSEL and other NSF Coalitions for fostering systemic curricular change can be termed a top-down approach to reform. Arguments can be made for and against a top-down approach to curriculum reform, in which faculty are involved in assessing an entire program of study and addressing needs in each component before implementation begins. At the bottom line however, using a top-down approach proves costly in faculty time, because these resources appear to be fixed. Hence levels of faculty buy-in sufficient to support systemic curricular change are difficult to achieve.

Top-down reform can be successful in two general contexts. If a college is a new entity, then developing systemic change becomes a degenerate activity; since there is no existing educational program, the focus becomes "doing it right the first time." An example can be drawn from a university experience in Spain. The School of Chemical Engineering at the University Rovira i Virgili in Tarragona, Spain went beyond the departmental level with their reform efforts to incorporate project-based cooperative learning teams. There, first-year chemical engineering students are involved in team design projects that are led by two fourth-year students. These projects are authentic, real-world tasks that require students to learn and integrate knowledge from calculus, chemistry, physics, fluid mechanics and a number of other courses. This reform transcended an individual department to include faculty from mathematics, chemistry, physics and other disciplines. Change management was critical for this reform and included a strong industry partnership with Dow Chemical Company as part of the faculty management education.[9]

The more common situation is that systemic reform is attempted within an existing college, set of departments, and engineering curricula. It is in this context that the NSF-supported consortia have operated, and have largely applied a top-down approach to the problem.

In the abstract, curricular reform would seem to be most cleanly and quickly achieved following a top-down approach. However, the revolutionary wholesale change in curricula entailed and the associated high level of faculty buy-in necessary to implement change before any change results are gathered and evaluated have resulted in little, if any, emulation of the systemic change programs crafted within the consortia contexts.

On the other hand, a more traditional approach to course-based reform focuses on change of a total curriculum by implementing change one class at a time. The seeming advantage of such "bottom-up" approaches is that change is implemented in an evolutionary manner, and thus the results of change would be observable piecemeal, and (assuming the change is successful) faculty consensus for change would be enhanced. In effect, this evolutionary change model would in the best case produce a "snowball effect" in developing faculty consensus that would favor change.

In addition to the slower rate of change of the bottom-up model, the underlying assumption that curricula reform can be affected by an accumulation of individual course adaptations is to-date unproven. The change goals need to have a more systemic focus than typically exists in many single course change efforts. A systemic orientation raises different issues in change processes that impede or support lasting reform. Fisher and Fairweather found, for example, that failure to recruit additional faculty to teach the target course (lack of incentives and rewards in the department); lack of fit with the curricula in other engineering departments; and institutional policies regarding student enrollment and declaration of major affected course-level reform efforts.[10]

If institutionalization of course-level reform is the goal, success requires taking into account a variety of factors beyond the individual course and department hosting it. Change efforts require knowing what works, what does not work and why, and being able to translate these findings into new circumstances.[11] Innovations are often not sustainable, and even when they are, more comprehensive understandings are required for those successful strategies to be adopted and for less effective measures to be avoided. The bottom line is that a more comprehensive approach to reform is required than that typically followed in single course change approach to bottom-up reform.

Several take-home messages follow from the above discussion that guided our selection of activities to best reach our goals for MATLAB integration.

- First, sustainable success for any program of systemic, curricular change is predicated on faculty support for the program.

- Second, top-down, all-at-once curricular change requires infusion of large amounts of resources, typically not available to most engineering collages.

- Third, although well motivated and although often successful in narrow terms, bottom-up attempts to change individual courses, do not usually work out to produce systemic change through an entire curriculum.

Our program to infuse pervasive use of MATLAB across engineering curricula recognizes and takes as starting points these three lessons. Like most schools of engineering, the second bullet applies to our current state. We believe the third bullet – that bottom up attempts to modify curricula have not gone beyond the single course level – is due largely because no research group has systematically tried to extend a bottom up approach to whole curriculum issues previously.

Finally, the first bullet – that faculty buy-in is key – is perhaps our most important starting point. It would seem to be "apple pie and motherhood" to say that faculty buy-in is key. Yet, again and again it has been shown that ambitious programs for curricular reform fail at the outset or (more likely) fail to be sustainable largely because faculty in general are not part of the process of change.

Specific Efforts at Michigan State University, College of Engineering

There is an accumulating and recent history of activities in the MSU College of Engineering that are part of our initial explorations to infuse MATLAB across our curricula. All of these efforts started with a recognition that following the traditional student path of study from the freshman year on was not working, viz.: MATLAB as the focus for a freshman level computation course, followed by a two-year break in student use of MATLAB, and ending with senior-level courses in which MATLAB proficiency was assumed.

Linkage between beginning MATLAB Course and Mechanical Engineering Course in "Statics"

In the summer of 2002 and 2003, a joint experiment was carried out involving students in the CSE 131 gateway course in MATLAB and engineering problem solving (CSE 131) and the beginning course in the Mechanical Engineering Department focused on mechanical systems in static equilibrium (then numbered ME 221). To help address the perception by many freshman and sophomore students that the beginning MATLAB course was a "hoop" to jump through, or a "weeder" course, students taking the courses during a summer session were commingled in work groups for final term projects, where the term projects were required course work for both courses.

Although the experiment yielded valuable insights into the attitudinal dynamics of the freshman and sophomore students[12] the experiment was not carried over into the mainstream for the regular academic year. The main impediment to implementation was found to be scheduling flexibility during the regular academic year. The main lesson from the experiment was that the data suggested by that attitudes for students in the MATLAB course towards MATLAB and its utility were positively affected by interaction with students from another (higher level) course in which MATLAB was used for problem solving. No definitive results were obtained because of small numbers of students involved in the experiment, although anecdotally the results were encouraging.

Specific take-home lessons include the following:[12]

- First, direct linkage between courses in the form of joint projects can shift attitudes of students on the importance of computational tools. (Tentative result only)

- Second, logistics in a large college without lockstep course scheduling will make it difficult to fully institute joint projects between courses.

Support for Experimental Course for Freshman Engineering

As part of overall efforts in our College of Engineering, the Mechanical Engineering (ME) Department is currently piloting a series of two courses (one year total) for freshman ME students. One of these courses parallels the current college-wide, gateway course that focuses on MATLAB. The instructor for the ME computation-focused course has been supported by direct linkage with personnel of the wider college course to help with problem set development. In addition, in the ME course has a common textbook with the wider college course in MATLAB.

The linkage between the ME-specific course and the wider college course has been in place now for one year. There have been a number of logistic and procedural lessons learned from this exercise which can now be generalized to develop like "people centered" support to enable development of MATLAB problem sets for class use.

Lessons learned from this experience predictably include the following:

- First, to encourage faculty acceptable of a new (to them) computational tool, direct support to help with problem set construction is very useful.

- Second, direct support of students by a knowledgeable TA to help answer MATLAB questions is an essential.

Linkage to the Chemical Engineering and Materials Science Department

As part of its compliance with ABET Engineering Criteria our Department of Chemical Engineering and Materials Science (CHEMS) regularly conducts assessment of student learning in the eleven ABET-designated program outcomes. During several customary assessment cycles, the faculty noted significant weaknesses in student performance in the application of math and effective use of engineering tools in the solution of engineering problems. These weaknesses were confirmed by feedback from alumni and cooperative education employers. When queried about this issue and as noted earlier in this proposal, the students pointed to large curricular discontinuities between the math and computer science that were taught early in the curriculum and their eventual application – or lack thereof – to the solution of chemical engineering problems in upper-level courses. Additional student testimony also indicated a direct linkage between the weak demonstration of math skills in problem solving and the inconsistent reinforcement and application of computational tools (MATLAB). Although upper-level courses did use a variety of engineering software including simulation software (ASPEN, Control Station) and computational tools such as Polymath, except for the use of EXCEL, most chemical engineering courses did not use MATLAB or any other computational tools taught in the beginning engineering problem-solving course (CSE 131).

In order to improve student learning and provide more continuity in this area, the CHEMS curriculum committee and the departmental faculty at large decided to 1) redesign the curriculum to include an applied mathematics course at the sophomore level and discard an upper-level transport phenomena course and to 2) incorporate the use of MATLAB throughout the curriculum. Three chemical engineering courses were selected as the test-bed for this endeavor. While seemingly a straightforward decision, it involved an added level of faculty commitment because a significant portion of the faculty were not regular users of or even familiar with MATLAB. In order for MATLAB to be applied in the upper-level courses, the faculty required education in the basics and potential advanced applications of MATLAB.

Through the individual efforts one of authors and direct College of Engineering support targeted by the College of Engineering Associate Dean of Undergraduate Studies, a MATLAB Quick-Start program was designed and presented to interested faculty during late August and early September 2005. Faculty participation was voluntary, but fully *forty percent* of the faculty members with regular undergraduate teaching responsibilities participated in the Quick-Start workshop.

The workshop consisted of hands-on participation in a computer lab, brief presentations by the workshop instructor, and short screen movies. To some degree, the workshop was self-paced. The workshop instructor included several example calculations taken directly from CHEMS course work that were supplied by CHEMS faculty. Topics ranged from MATLAB basics to more advanced topics tailored to potential classroom applications.

Individual on-site tutoring was also provided. A key feature of the success of the Quick-Start workshop was that knowledgeable teaching assistants assigned with seed funding from our college supported the faculty who applied MATLAB in their courses. These TAs could help with details of implementation. For the 2005-06 academic year, MATLAB was initially incorporated into three chemical engineering courses: ChE 210 (Modeling and Analysis of Transport Phenomena), ChE 312 (Mass Transfer and Separations), and ChE 432 (Process Control) thus providing a MATLAB experience integrated in sophomore, junior and senior level courses.

Important take home lessons from one year piloting of support for MATLAB infusion into CHEMS includes the following, and were discussed at ASEE in June in the more narrow context of chemical engineering.

- First, departmental buy-in to the importance of MATLAB applications across the undergrad curriculum is centrally important – Curriculum Committee buy-in and Chair buy-in is a prior because of the leadership roles both play.

- Second, a faculty "quick start tutorial" is very useful if not essential in terms of kick off activity, in part because it demonstrates that faculty unfamiliar with MATLAB will get direct support when and in the amount they require it.

- Third, direct TA support for problem set development, and for helping students with MATLAB problems is important for success.

Quantitative data on the CHEMS experience can be found in a companion 2007 ASEE paper: "Course Assessment for Curricular Reform" by Briedis et al.

Initial Prototype for Web Resource to Support MATLAB Infusion

With direct and indirect support from our College of Engineering, in the form of a small seed grant and a new and very capable server for the web, a small prototype MATLAB resource was developed for use with the CHEMS Department Faculty Quick Start, and to support students and faculty over the 2005-2006 Academic Year in the courses in which MATLAB was infused. The small beginning web resource was oriented at (a) beginning MATLAB topics, and (b) a limited number of specialized topics, chiefly on numerical solution methods for differential equations to support CHEMS courses.

Content of the web resource has so far been limited to screen movies that are oriented at the rudiments of the MATLAB topic addressed in each "lesson." Initial experience with the web resource has reinforced the belief that a "how to" resource, generally accessible to students, and directly supporting course work in target MATLAB infusion courses will help to alleviate pressure on instructors who are themselves just learning MATLAB.

From our experience in 2005-2006 with producing a web-available resource to support MATLAB, we have learned the following:

- First, the screen movie web resources should be relatively short, or segmented so that students can "try out" newly learned capability easily. I.e., the student should be able to learn a narrow facet from a screen movie (or segment of one) then turn to MATLAB and try the capability. To support this idea, examples the students can try are useful.

- Second, testing for the screen movies should be phased and relatively rigorous. (Our experience is that a "screen movie" with a "typo" convinces students to not use the screen movie resource at all, ever again.)

More recently, the MathWorks, implementers of MATLAB have given support to our project to infuse MATLAB in our curricula, and in particular to develop web resources that will be shared both across our college, and made available generally.

Summary and Future Directions

We are embarked on a path to reach an ambitious goal: to effect systemic curricular change across our engineering undergraduate programs resulting in our graduates being self reliant in the use of MATLAB. Institutionally, the same goal can be phrased as developing pervasive computing as an integral part of our undergraduate engineering programs. We were lead to our current state not by a overarching desire to effect curricular reform, but rather by a need we saw to better integrate computational tool use into our programs in such a way that students truly do become self sufficient in the computational tool use. Following our initial discussions of how to

reach our targeted goal for MATLAB integration, we were naturally brought to the very broad subject of systemic curricular reform.

Our approach, rooted in our specific problem, can be characterized as *evolutionary* curricular change as opposed to the more prevalent model that could be term *revolutionary* curricular change. We seek to effect change in small steps that over time result in large-scale program shift.

The core of our method to effect the change we seek is a multi-step process:

- to select linked islands of opportunity (classes) within a given program,
- to engage the instructors for the identified classes in a process to learn MATLAB (if necessary), and
- to support the instructors as fully as possible during their initial offerings with MATLAB and on a continuing basis
- to support student use of MATLAB by developing a solid "How To" web-based resource.

Currently our specific test bed lies largely in our Chemical Engineering program. Results are beginning to flow that we find encouraging. In addition to initial formal results (as detailed in the ASEE 07 paper "Course Assessment for Curricular Reform" by Briedis et al, anecdotes are also accumulating. Lead faculty in our gateway MATLAB course have noted a sense among students that the course is just a "hoop" and serves no purpose in their later coursework. The situation was that MATLAB was used in the curricula, but not until junior or in some case senior years. The gap in use lead to a student culture of believing that MATLAB was "worthless."

Within the Chemical Engineering program, students have radically shifted their general mindset. In that program where now three courses systematically use MATLAB – from freshman level up – the students recognize the gateway course in computational tools as being at least as important to them as their calculus sequence. As an indication of this shifting mindset, consider the following. The gateway course employs a number of undergraduate lab mentors to help students in their formal lab meetings. This year, with the Chemical Engineering effort to integrate MATLAB through the entire curriculum, the percentage of undergraduate lab mentors in ChE has risen from about 20% to about 70%. When asked why the change had taken place, a common response from the lab mentors in ChE was that "We know that the department expects us to know MATLAB and being a lab mentor is a great way to be extend our learning." The message has been given – the student culture in ChE has shifted.

Our near term efforts will be to continue support for ChE faculty for the use of MATLAB in their courses, to work closely with ChE faculty (Curriculum Committee and Chair in particular) to expand our number of covered islands, and to continue formal studies of the effect of the shift to integrating MATLAB in our ChE Department.

Long term, we hope to initiate a longitudinal study tracking the progress of ChE students up through the ChE curriculum specifically on their competency and attitudes towards

computational tool use, and its impact on the freedom on instructors to engage more in concept level issues in their classes. Going further, once we are sure of our ground based on our experience with one department, we hope to move on to apply our lessons to our entire college.

Acknowledgement

We gratefully acknowledge a gift from the MathWorks. This gift will enable continuation of our current efforts to develop and sustain systematic use of computational environments vertically in our undergraduate engineering majors.

References

[1] Stoll C. When slide rules ruled. Scientific American. 2006 May, 2006;294:80-7.
[2] Cotton K. Teaching Thinking Skills. School Improvement Research Series (SIRS) [web access] 1991 [cited January 12, 2007]; Available from: http://www.nwrel.org/scpd/sirs/6/cu11.html
[3] National Research Council. Improving engineering design: Designing for competitive advantage. Washington, D.C.: National Academy Press 1991.
[4] Bordogna J. Making connections: The role of engineers and engineering education 1997.
[5] National Science Foundation. Shaping the future: New expectations for undergraduate education in science, mathematics, engineering, and technology. Washington, D.C.: National Science Foundation 1996.
[6] Bordogna J, Fromm E, Edward E. Engineering education: Innovation through integration. Journal of Engineering Education. 1993:3-8.
[7] Hall SR, Waitz I, Brodeur DR, Soderholm DH, Nasr R. Adoption of active learning in a lecture-based engineering class. In: Bjedov G, editor. Frontiers in Education; 2002 November 7; Boston, MA: IEEE; 2002. p. T2A_9-15.
[8] Terenzini P, al e. Students' out-of-class experiences and their influence on learning and cognitive development: A literature review. Journal of College Student Development. 1996;37(2):149-62 (EJ 527219).
[9] Witt HJ, Alabart JR, Giralt F, Herrero J, Medir M, Fabregat A. Development of coaching competencies in students through a project-based cooperative learning approach. In: Bjedov G, editor. Frontiers in Education; 2002 November 7; Boston, MA: IEEE; 2002. p. F2A_1-6.
[10] Fisher P, Fairweather J, Amey M. EC2000 and organizational learning: Rethinking the Faculty and Institutional Support criteria. Annual Meeting of the American Society for Engineering Education; 2002; Montreal; 2002.
[11] Senge P, Associates. The Fifth Discipline: The Art and Practice of the Learning Organization. New York: Doubleday 1990.
[12] Hinds T, Amey M, Eskil MT, Sticklen J, Urban-Lurain M. Curricular Integration of Computational Tools: A First Step. *2005 ASEE* 2005.