2006-1238: ALGORITHMIC THINKING AND MATLAB IN COMPUTATIONAL MATERIALS SCIENCE

Paris von Lockette, Rowan University

Paris von Lockette is an Associate Professor of Mechanical Engineering at Rowan University. He received his Ph.D. from the University of Michigan, Ann Arbor in 1999. His interests include the physics of polymers and numerical / computational methods in materials science.

Algorithmic Thinking and Matlab in a Computational Materials Science Course

Abstract

A course was developed to teach aspects of materials science, numerical methods, and programming in an integrated fashion. During the second teaching of the course, it was modified to enhance its delivery by focusing on the aspects which gave the students the most difficulty in its first offering: syntax and organization of operations in programming. This was achieved through the use of Matlab as a meta-language platform, development of Matlab tutorials for the course, and an emphasis on algorithmic thinking.

In this paper, algorithmic thinking involves developing a complete understanding of the operations required via hand calculations and block diagrams *before* attempting to generate any code. Students were graded on their ability to relate what the program/algorithm should do next verbally and pictorially and then tasked with translating those known operations into Matlab code using Matlab's extensive help menus. The help menus allow users to employ keyword searches to find descriptions and examples of commands with the needed functionality.

Results of student projects show improvement from the first to second years. Student response to the course also shows an increased respect for Matlab as a useful engineering tool. In both years, students who were unable to verbally describe the needed operations in the programs generated less efficient or inoperable code.

Introduction

Computational Materials Science (CMS) is a cross-listed senior elective and graduate course in Mechanical Engineering that meets for 75 minutes twice weekly. The course is also part of a newly created Materials Science Concentration. The course covers topics in three fundamental areas: numerical techniques, geometric and potential-energy aspects of materials science, and programming. Numerical techniques primarily involve minimization of potential functions while the materials science concepts center around Lennard-Jones potential functions and the network structure of polymers. The course is offered biannually and was last offered in the Fall 2004-2005 academic year. During this last teaching, the course was revised to better facilitate student comprehension and application of concepts based on observations from previous teachings. The content of the 3 semester hour course is broken into roughly 40% numerical techniques, 25% materials science, and 35% programming using the text: Andrew Leach, Molecular Modeling: Principles and Applications[3].

While students in mechanical engineering are well versed and comfortable with the *theory* behind numerical minimization techniques ("if df/dx=0 then you are done") and the basic theories behind materials science concepts ("the structure wants to achieve a minimum energy"), they are quite uncomfortable with programming. Furthermore, they rarely, at the undergraduate

level, are given the opportunity to see explicit links between the material science and numerical minimization. This course attempts to draw students into the intersection of numerical techniques, materials science, and programming by removing, as much as possible, the programming aspect that causes students the most difficulty: syntax. The course relies heavily on Matlab and the students' ability to think through what a given algorithm should do to aid them in bridging the gap between the three areas. To achieve this, formal graded emphasis was placed on teaching students to think and work through the rote operations of numerical techniques before attempting to generate programs. The goal was to generate, as Connolloy put it, a "Prior Knowledge Environment"[1] wherein students had thorough prior knowledge of a technique before trying to translate that knowledge into programming language to solve numerical problems in materials science.

Students in our mechanical engineering program are required to have had a computer course prior to entering college and are also required to take an entry level computer programming course during their first year (though many test out of this requirement). These courses do not teach matlab and cover only rudimentary topics such as *if-else* statements, *for/while* loops, and assignments to variables and matrices. It is only recently that the computer science department has moved from teaching Visual Basic to teaching C++. By the time our students reach their senior years they have had only what programming has been incorporated into projects or assignments throughout our curriculum. The author knows of only one course on our curriculum below the senior level that engages the students in significant programming, hence they are not well versed in programming upon entering the course and often do not remember using Matlab.

This paper will begin by giving a cursory overview of the numerical methods involved, emphasizing how they are presented to students, followed by example assignment used to address these concepts. Next, the links between Materials Science and Numerical Techniques are discussed, again with an emphasis on how they are presented to students. Finally, the schema for incorporating programming is introduced. The paper ends with some qualitative observations and comparisons between the academic years.

Numerical Techniques

In this course, the potential functions, Π, needed for modeling material structures can be cast in a form similar to

$$(1) \qquad \Pi(x_1, x_2, \mathrm{K}\ , x_N) = \sum_{j=1}^{N} \sum_{i=1}^{N} f(x_j, x_i, c_i, c_j)$$

where x's are independent variables, c's are constants, and $f(x_i,x_j,c_i,c_j)$ is some function which operates on paired combinations of the entire set of independent variables. After introduction of this concept, initial class discussion is focused on how one would find the minimum of such a function. The visual marker $f(x)$ leads students back to calculus and the familiar methods of setting $f'(x)=0$ as a starting point. Initial in class problems also tasked students with using this approach to find the minimum of a simple function,

$$(2) \qquad f(x_i,x_j)=(c_ix_i-c_jx_j)^2$$

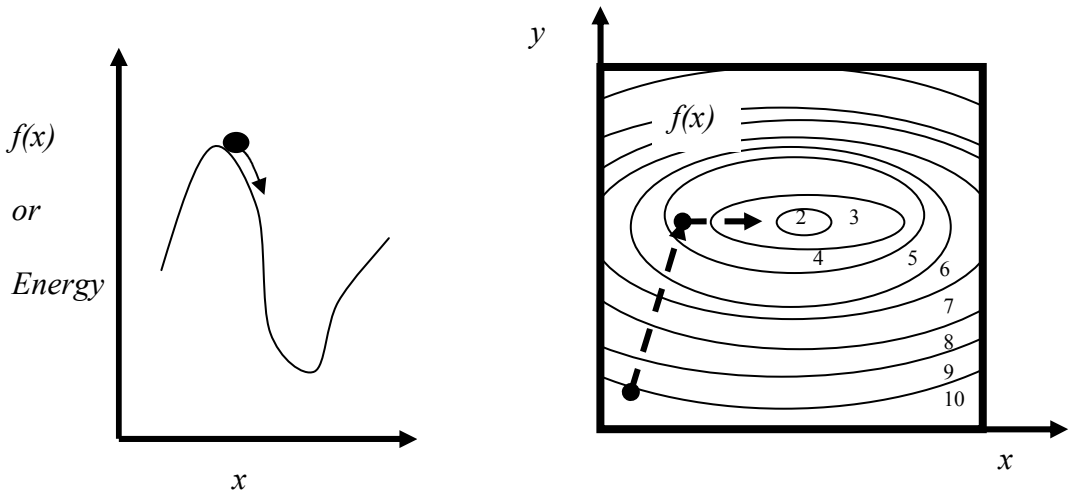

Figure 1: A schematic of a 1-D case highlighting how a ball will always roll down hill (left). A line minimization still moves to the bottom of the hill (right) but may not hit the very bottom on the first pass.

by hand when the constants are all given. For $N=2$ the problem reduces to a parabolic equation which the students recognize after they have performed the summation in (1). They can solve this readily by hand. After performing the summation for $\Pi(x_1,x_2,x_3)$ the problem begins to loose physical meaning for undergraduates (though some recognize it as a surface in three dimensions) but the notion of taking derivatives still holds and they are able to generate a system of equations to solve after some instruction. Next, the leap is made to the case of $N=8$ which would yield Π having 64 terms in the most general case of (1). Students are asked in class to start taking derivatives and generating the list of equations to solve up to the point that someone asks, or suggests that, there might be a better way.

At this point students understand the need for taking derivatives, fore evaluating the potential function, and for generating a list of equations to be solved strictly from the concept that $f'(x)=0$ at an inflection point even over a possibly unlimited number of dimensions, e.g. as N goes to infinity. Other concepts of linear algebra such as sparse matrices are also addressed but not commented on here. Next, formal derivative- and non-derivative based numerical minimization methods are discussed.

Discussion of Various Solution Techniques

Students are familiar with minimization using derivatives from mathematics courses so these serve as a natural launching point for discussion of derivative-based numerical methods. The ubiquitous "ball rolling down the hill" example is used to visually convey the gradient philosophy, "always go down hill" (see Figure 1, left). Next, conjugate gradient methods are discussed with emphasis placed on line minimizations. Line minimizations help bridge the gap between minimization of functions with one independent and one dependent variable (e.g. $y=f(x)$) and multivariate minimization techniques highlighting, visually, the fact that the function is still being minimized in only one "direction" (see Figure 1, right).

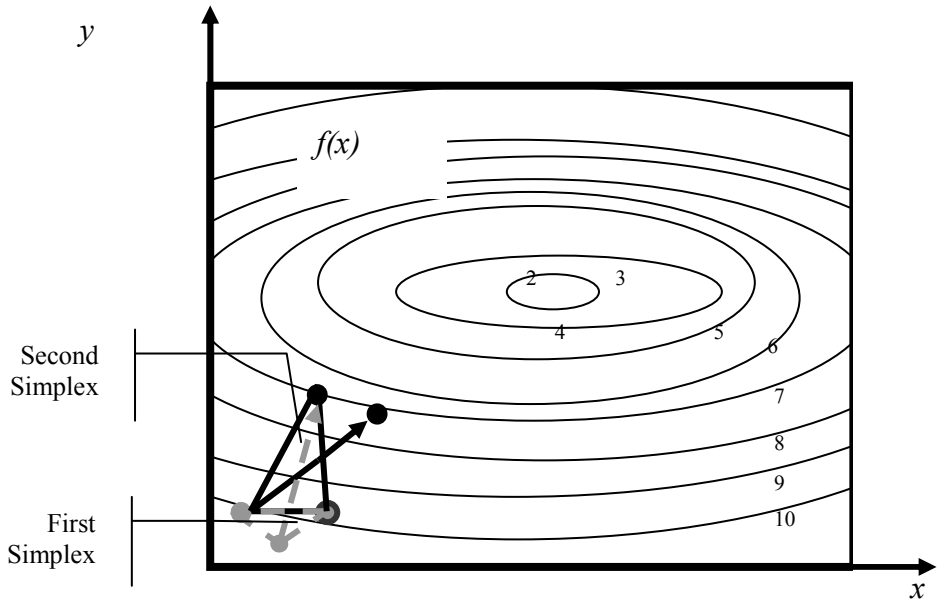

Figure 2: Schematic of two successive iterations (simplexes) in the Amoeba (or Downhill Simplex) method. The first is in gray, dotted lines. The second is in dark black lines

Students are given the equation for $\Pi(x_1, x_{2,,})$ of Figure 1 and the equation of a line passing through the Π surface. Then they are asked to estimate the minimum of $\Pi(x_1,x_2)$ along the line first by plotting values, then to calculate it explicitly by using calculus when c's are known.

At this point, the core concept of derivative-based methods, namely line minimizations, has been covered analytically and pictorially. Students may not necessarily understand the mechanics of the conjugate gradient's method of choosing parameters surrounding the length of the line segment and other factors, but the notion of minimization beginning at a point and continuing along specific directions within the variable space, directions which may not necessarily lead directly to the global minimum, has been conveyed.

Introduction to Algorithmic Thinking

Several other non-gradient methods including the Metropolis algorithm[2] and the so called Amoeba[2] method are discussed at this point in the course. These methods dovetail into the notion of numerical minimization techniques that start at a given point and travel in directions within the variable space that may not necessarily lead directly to the global minimum. In addition, these methods begin to veer away from student's familiarity with the concepts and into the realm of using algorithms instead of direct calculus to solve minimization problems. Class discussion begins with the Amoeba method since it is similar in appearance to line minimizations.

Pictorially the Amoeba method looks similar to the line minimization methods, save for the fact that it is constantly changing directions. Algorithmically, the method evaluates the function at

N+1 points, then draws a line from the highest valued point through the centroid of all *N+1* points. The length of the line is arbitrarily set. In this fashion, the method steps from a high-valued point to a lower-valued point successively. As with the previous line minimizations, students are tasked with performing the algorithm by hand given the function Π for the variable space and a set length for the Amoeba step with *N=2*. It quickly becomes apparent that the method has a very slow rate of convergence and that for N → infinity, the calculations become numerous. Students also realize that it is not necessary to calculate derivatives, only evaluate the function.

The Metropolis algorithm is a novel technique that seeks to minimize a function by varying the independent variables either randomly or in some preset method then either accepting or rejecting the new values based on probabilistic criteria. This algorithm is the farthest from the students prior knowledge but builds on the concepts of using function evaluations based on values of the independent variables and the idea that it is not necessary to move directly toward the global minimum since it is usually unknown. Algorithmically, Metropolis chooses an initial set of values for all *N* variables, evaluates the functional in this configuration, then chooses a new set of variable values and either (1) automatically accepts the new configuration if the functional value was reduced, or (2) has the probability, *p,* of accepting the new configuration. The probability has the form

$$p = \exp\left[-\frac{\phi(\Pi)}{\tau}\right] \tag{3}$$

where $\phi(\Pi)$ is related to the value of the functional and τ is a scaling factor which adjusts the acceptance rate. In this fashion, new choices of variables which do not lead toward a minimum can be accepted. In class discussions this concept is related to thinking outside the box and, more importantly, related to the primary flaw of conjugate gradient methods, e.g. they can get caught near local minimums and never find the true global minimum.

At this point in the class the students have been exposed to several numerical solution techniques for multivariate problems and the mechanics of the operation of each method. Strengths and weaknesses of each method are also discussed. In addition to hand-calculations on exams, students are asked to explain the methods in annotated sketches and by giving written descriptions of the operations. Looking at the responses for all methods, the pictorial representations showed better overall understanding of the various methods than the written descriptions.

The next segment of the course is the linchpin that introduces the concept of algorithmic thinking which is defined as *a repetitive sequence of operations*. Looking back at the solution methods, students were asked to formulate the basic operations that make-up each method with respect to the objective function Π. Eventually the discussion distills the methods down to the operation given in Table 1.

Table I: Steps in the Completion of Numerical Minimization

Step	Description

1	Choose a configuration: this may be an initial configuration (i.e. values for all variables) or the next configuration in an iteration.
2	Evaluate the functional: calculate the value of the functional in the configuration.
3	Evaluate the derivative of the functional: calculate the derivative with respect to an independent variable (possibly all independent variables); some methods do not require this step.
4	Accept or reject the configuration: most methods accept automatically save for Metropolis which places additional criteria on the new configuration.

In this coarsely grained description the process of numerical minimization is reduced to three or four necessary operations. The mechanics of the operations are preset either by the functional (steps (2) and (3)) or by the method (steps (1) and (4)). In terms of programming, the key is to find a tool which allows students to translate these operations into code with as little difficulty as possible. The next section details the link between minimization and material science concepts and is followed by the culmination of linking these aspects to Matlab and programming.

The Lennard-Jones Potential Function in Materials Science

An introductory concept in Material Science is that of the Lennard-Jones (LJ) potential function which gives the energy associated with the charge interaction of two atoms, see Figure 3. A standard form of the LJ potential function is given by

$$\Pi\left(\bar{x}_i, \bar{x}_j\right) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{4}$$

where

$$r_{ij} = \left\|\left(\bar{x}_i - \bar{x}_j\right)\cdot\left(\bar{x}_i - \bar{x}_j\right)\right\| \tag{5}$$

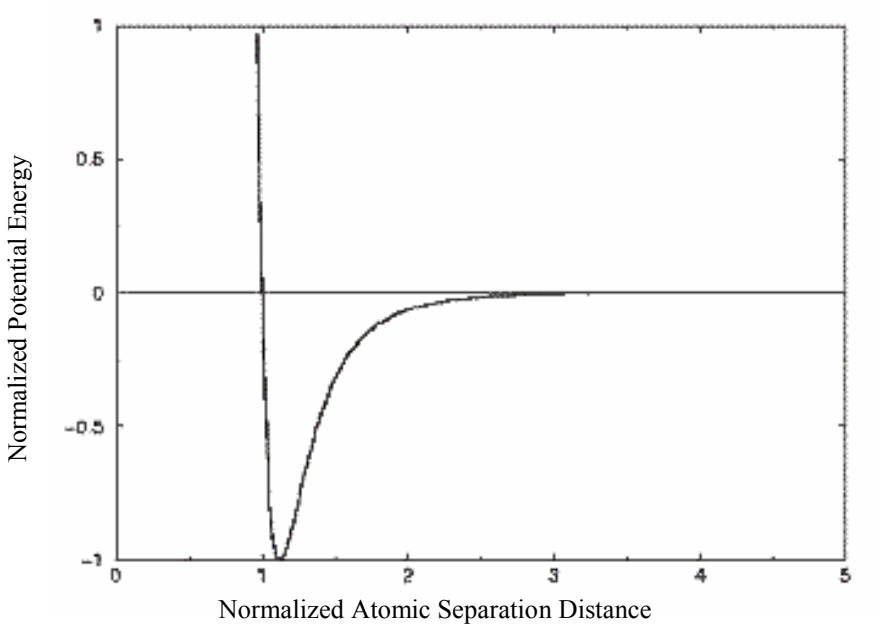

Figure 3: Lennard Jones Potential Function

for the interaction between two atoms at positions $\mathbf{x}_i$ and $\mathbf{x}_j$ separated by distance r_{ij}. The constants, σ and ε represent the equilibrium spacing and energy well, respectively. When more atoms are present, the LJ potential operates pair-wise between all atoms requiring summation to find the total energy. The forms of (4-5) are quantitatively similar to that of (1).

In the sophomore level materials science course the LJ potential is used to highlight the idea that pairs of atoms reside at an equilibrium spacing that minimizes the potential energy between them. When introduced in concert with a simple cubic lattice structure such as a BCC crystal it becomes clear to students that the number of calculations is too large for pen and paper. The fact is even more evident when considering even low molecular weight polymer chains. A full analysis requires *N^2-sum(0…N-1)* evaluations of (4-5) which would yield 10 pairings for the BCC structure and 36 pairings for simple polyethylene molecule with only three backbone chains.

The method of finding the equilibrium spacing for a set of atoms is qualitatively and quantitatively similar to finding the minimum of the basic functions discussed earlier in (1) and (2). Students are presented with the fact that

$$\frac{d\Pi}{dx_i} = 0 \;\forall\; i = 1\text{K}\;\; N \tag{6}$$

forms the basic set of equations to be solved which is seen as tractable when N is small. This is equivalent to minimizing the functional, Π, itself.

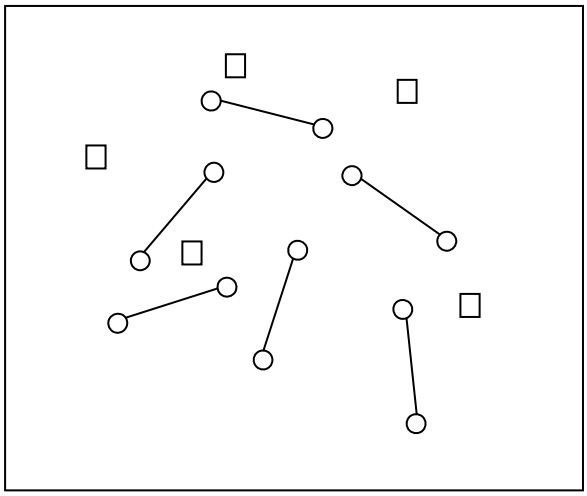

Figure 4: Virtual reaction space with chain-ends (circles) and reactive agents (squares)

At this juncture, the concept of the algorithmic approach reintroduced as a means to solve the problem. The steps are shown to be the same: choose x_i's, evaluate Π, choose new x_i's, and either keep or reject the new configuration. To place the focus on algorithmic thinking, e.g. repetitive operations, students are tasked with utilizing the Ameoba and metropolis algorithms, first by stepping through the algorithms by hand calculations for small systems before programming methods are introduced.

Simulations of Polymer Network Formation

Some simplified computational models the formation of polymer networks begin by seeding a virtual reaction space with randomly arranged points that represent chemical crosslinks and randomly arranged line segments which represent individual polymer molecules[3,5 – 9]. In the simulations performed for this class, a chain end was allowed to bond with a reactive agent only once while reactive agents could join with up to four chain ends. This simulation contains difunctional polymer chains and tetrafunctional reactive agents. Simulation of polymerization, the actual linking of the molecules and the crosslinking agents, is carried out in a nearest neighbor fashion. The nearest neighbor approach seeks to find *the closest crosslink,* a*, to chain end,* b*, that is itself not closer to any other crosslink.*

Students are first given a set of random *x-y* points representing crosslink and chain end positions. Next they are given the problem of determining the crosslinks which are nearest neighbors of each chain end. Students quickly see that the problem, though simply stated, is nontrivial. In fact in computer science fields it is classified as an intractable problem since the number of operations required to solve it cannot be reduced or truncated. The nearest neighbor search is introduced as the final problem for the course since it moves away from calculus-based mathematics altogether and focuses on more fundamental repetitive operations, searches and value comparisons.

In terms of an algorithm, the nearest neighbor search requires the steps in Table II.

Table II: Steps to Completion of Polymer Network Simulation

Step	Description
1	Generate random numbers to seed the reaction space.
2	Calculate the distances between all reactive agents and chain-ends in the virtual reaction space.
3	Sort and store the distances.
4	Find the reactive agent, *k,* closest to chain end *i* and check to see if *k* is closer to any other chain-end than it is to chain-end *i.*
5	If agent *k* is closest to chain-end *i* and vice versa, then the two may bond together.
6	*if* reactive agent *k,* has a bonding site available, reduce the number of available sites on *k* by one and record its link with chain-end *i.*
7	If (4) through (6) are true, then remove chain end *i* from the list of available reactive chain ends and return to (4) looking at the next chain-end, incrementing *i=i+1,* until all chain-ends or all reactive agents are used.

Students understood the methodology well enough to perform the operations by-hand as evidenced by homework, but generating code in a programming language to perform the tasks was more challenging. Syntax is usually the largest barrier by students to learning programming languages. Consequently, even when all the required operations are understood, utilizing a programming language to perform them is still daunting. To facilitate turning existing knowledge into a programming language, students were given instruction in Matlab, a command-line, interpreted language use by scientists, engineers, and mathematicians.

Programming

Matlab is an interpreted or meta language which can be run from either a command line or from 'batch' files. Matlab supports thousands of commands to perform both basic algebraic functions such as adding and subtracting, as well as much more powerful functions such as sorting and searching. In addition, Matlab programs require little or no variable declarations or sizing of arrays. A powerful feature of the newest version of Matlab is its help functions which allow the user to search for commands by keyword. The help files contain formal definitions of functions as well as examples of their use. With Matlab's broad functionality an integrated help functions, users can search for predefined functions that complete the tasks they require; one can usually find an example that shows a relevant implementation of the function as well. The program also checks commands for syntax, often providing guidance on how to fix mistakes.

In the programming section of the course, students were given two large projects. The first was the generation of polymer networks and the second was the annealing of those networks into equilibrium structures. The annealing process was simply the minimization of the energy of the network when each polymer was treated as a nonlinear spring having a stored energy equal to

$$(7) \qquad \Pi(\bar{x}_1, \bar{x}_2 \mathrm{K}\ \bar{x}_n) = \tfrac{1}{2} G \sum_{i=1}^{n} \sum_{k=1}^{4} 3 d_{ik} + \frac{9}{5} d_{ik}^3 + \frac{297}{175} d_{ik}^5 + \frac{1539}{875} d_{ik}^7 + \frac{126117}{67375} d_{ik}^9$$

where

$$(8) \qquad d_{ik} = \frac{\sqrt{(\bar{x}_i - \bar{x}_k) \cdot (\bar{x}_i - \bar{x}_k)}}{d_{\max}}.$$

Equations (7) and (8) give the energy in each chain according the freely jointed chain model which uses the length of a chain with ends at $\mathbf{x}_j$ and $\mathbf{x}_k$ in a truncated inverse Langevin series before summing over n reactive agents[4]. The rubbery modulus, G, summation from 1 – 4, and the maximum chain length, $d_{\max}$ are constants associated with the type of polymer being simulated. The form of the energy is identical to (1). These two projects were direct applications of the topics covered earlier in the course.

To start off the projects the class discussed the operations needed to complete the steps in Tables 1 and 2 which included: *adding, subtracting, raising to a power, generating random numbers, sorting arrays, storing arrays,* and *indexing arrays*. In the larger context, the first project, forming the network, was necessary to complete the second, annealing the network. Next, students where given a "Boot Camp Matlab Tutorial" which performed step (1) of Table 2 utilizing commands to perform all eight operations discussed. The assignment forced students to reverse engineer the Matlab code in order to understand the uses of each function and then tasked them with performing steps (2) and (3) using those commands. The tutorial was purposefully made hard.

The progression toward this assignment was to first teach students how to perform the mechanics of the operations involved in each simulation by hand calculations, next to place those operations into context of the larger algorithm, and finally, once they had an understanding of the problem as evidenced by homework and exams, to immerse them into the problem *in another language*. The author would suggest that the methods are similar, at least at a surface level, to the way in which immersive verbal language courses operate. One is placed into familiar situations but forced to communicate in an unfamiliar language. The final four weeks of the course were spent working on the two large projects. During that period class time was spent each week discussing what commands and functions students had found in Matlab that were useful to the project, similar to learning another language by speaking it regularly.

An important series of assignment also tasked students with generating block diagrams of their programs which incorporated variable names and detailed what operations were required, their timing within the program, what information was required in the operation, and what variables were generated or modified. During the 2002 teaching of this course it was seen that a failure to clearly understand these relationships early on caused students either to generate inefficient algorithms that attempted to account for early lapses in logic in an *ad hoc* fashion or would lead to their inability to generate the solutions altogether. Students who successfully generated the complex methodology on paper were better able to translate that into Matlab pseudo-code.

Results, Assessment, and Observations

Several factors point to the overall success of the methodology used in the 2003 teaching of the course over the standard methods used in the 2001 teaching.

- Eighty percent of students completed both projects successfully in the 2003 teaching of the course as compared to less than half the students in the 2001 teaching. In the 2001 teaching formal, graded emphasis was not placed on thinking through complex algorithms before attempting to generate code whereas 2003 included this at several stages.
- Response to the tutorial was exceptional. Several students commented that they wished they had learned Matlab in this fashion earlier in their college careers.
- Two-thirds of students engaged in self-guided learning of programming language as evidence by a special challenge assignment. Students where challenged with generating Matlab code to perform the computational simulation of polymer formation in as few commands as possible. Two thirds of students developed functioning programs utilizing fewer commands than the professor's. This was accomplished by finding Matlab commands which combined necessary functions.
- During and following the course 1/3 of all students commented positively on the course in communication with the professor. Comments included, "I now see Matlab as a useful tool", "I can use this in my other courses", "I wish I had learned this for/in my other courses", and "I may never use this[Matlab] again, but I could if I had to".

Issues that were present in the 2001 teaching of the course also reoccurred in the 2003 teaching.

- Students with good block diagrams generated more streamlined programs.
- Students who did not complete correct block diagrams generated inefficient (spaghetti) or inoperable code. Their programs attempted to correct for lapses in initial organization by "tacking on" fixes which rarely worked.

References

1. Creating a Prior Knowledge Environment to Facilitate the Introduction of Computer Languages and Programming to Engineering Students, ASEE Gulf Southwest Section Annual Conference, College Station, TX, March 2001.
2. William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling (1992) Numerical Recipes in C, Cambridge University Press; 2 edition.
3. Andrew Leach Molecular Modelling: Principles and Applications Prentice Hall; 2 edition (March 30, 2001)
4. Treloar, L. R. G., The Physics of Rubber Elasticity, (1975), Oxford University Press, Oxford, U.K.
5. Eicheinger B. E. and Leung, Y. K. (1984) Computer Simulation of End-linked Elastomers. I. Trifunctional Networks Cured in the Bulk , *Journal of Chemical Physics*, **80**, 3877-3884.

6. Eichinger B. E. and Leung, Y. K. (1984) Computer Simulation of End-linked Elastomers. II. Bulk Cured Tetrafunctional Networks, *Journal of Chemical Physics*, **80**, 3885-3891.
7. Hagn, C., Wittkop, M., Kreitmeier, S., Trautenberg H. L., Holzl, T., and Goritz, D. (1997) The Creation and Spatial Structure of End-Linked Bimodal Polymer Networks: A Monte Carlo Study, *Polymer Gels and Networks*,**5**, 327-337.
8. P. R. von Lockette and E. M Arruda, "Topological Studies of Bimodal Networks," *Macromolecules* (1999) **32**,1990-1999.
9. Termonia, Y. (1990) Molecular Model for the Mechanical Properties of Elastomers. 1., *Macromolecules*,**23**.