December 2010

Volume 1 Issue 1

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Volume 1, Issue 1 **December 2010**

JOCSE

Journal Of Computational Science Education

The Journal Of Computational Science Education (JOCSE), ISSN 2153-4136, is published quarterly in online form, with more frequent releases if submission quantity warrants, and is a supported publication of the Shodor Education Foundation Incorporated. Materials accepted by JOCSE will be hosted on the JOCSE website, and will be catalogued by the Computational Science Education Reference Desk (CSERD) for inclusion in the National Science Digital Library (NSDL).

Subscription: JOCSE is a freely available online peer-reviewed publication which can be accessed at `http://jocse.org`.

Contents

Introduction to the First Issue

Steven I. Gordon
Editor
Ohio Supercomputer Center
Columbus, OH
sgordon@osc.edu

Purpose of the Journal

It is with great pleasure that we release the first issue of the Journal of Computational Science Education. The journal is intended as an outlet for those teaching or learning computational science to share their best practices and experiences with the community. Included are examples of programs and exercises that have been used effectively in the classroom to teach computational science concepts and practices, assessments of the impact of computational science education on learning outcomes in science and engineering fields, and the experiences of students who have completed significant computational science projects. With a peer-reviewed journal, we hope to provide a compendium of the best practices in computational science education along with links to shareable educational materials and assessments.

Inaugural Issue

In this issue, we have a variety of articles that reflect the breadth of practices in the developing interdisciplinary computational science field. The articles by Parker and Shafii-Mousavi and Kochanowski provide two views of mathematics education using computational science. Parker describes the application of Computational Algebraic Geometry algorithms to teach both the mathematics and application of mathematics to problems in science and social science. Shafii-Mousavi and Kochanowski show how using Excel to address a range of service learning projects can be successful in teaching mathematical concepts and engaging students while they apply mathematics to practical problems.

The article by Smith et.al. provides an application of computational science to the high school curriculum to help chemistry students visualize the chemical structure of the ingredients in energy drinks and compare them to fluorescent chemicals.

The article by Sendlinger and Metz describes an overview of a series of workshops on computational chemistry and the materials assembled to assist faculty with introducing computational methods in their classrooms.

Shiflet and Shiflet present two modules that involve high performance computing and their application to two biological modeling questions across three different classes. The module concepts, as well as impacts of the materials on student understanding of both the scientific applications, mathematics, and algorithms applied to the models are discussed.

Finally, the two student articles provide their perspectives on the application of computational science to two very different problems. They describe the results of their projects along with reflections of its impacts on their learning and advice for future students and their mentors.

Acknowledgements

The efforts of many people have made this journal possible. First I must acknowledge the assistance of the editorial board. They have helped to define the role of the journal, the review process, the format, and, in many cases, the content of the journal. Assisting them were a number of reviewers for the submitted articles.

Next, I must acknowledge the key role of the Shodor Education Foundation and its staff. They have provided the mechanisms for the submission, review, and publication of the journal. Special thanks is due to both Dr. Robert Panoff for his support and leadership in creating the journal and to Jennifer Houchins who oversaw and completed the website and the back end tools that make the journal possible.

Finally, the National Science Foundation must be acknowledged for funding the creation of the Computational Science Education Reference Desk and the extension of that NSDL project which has helped us to start the journal. This was done under grants DUE-0435187 and DUE-0937910. Any opinions or other content are of course the responsibility of the journal or the authors and are not those of the NSF.

Computational Algebraic Geometry as a Computational Science Elective

Adam E. Parker
Department of Mathematics and Computer Science
Wittenberg University
P.O. Box 720
Springfield, OH 45501
aparker@wittenberg.edu

ABSTRACT

This paper presents a new mathematics elective for an undergraduate Computational Science program. Algebraic Geometry is a theoretical area of mathematics with a long history, often highlighted by extreme abstraction and difficulty. This changed in the 1960's when Bruno Buchberger created an algorithm that allowed Algebraic Geometers to compute examples for many of their theoretical results and gave birth to a subfield called Computational Algebraic Geometry (CAG). Moreover, it introduced many rich applications to biology, chemistry, economics, robotics, recreational mathematics, etc. Computational Algebraic Geometry is usually taught at the graduate or advanced undergraduate level. However, with a bit of work, it can be an extremely valuable course to a sophomore student with linear algebra experience. This manuscript describes Math 380: Computational Algebraic Geometry and shows the usefulness of the class as an elective to a Computational Science program. In addition, a module that gives students a high-level introduction to this valuable computational method was constructed for our Introductory Computational Science course.

Keywords: Gröbner Bases, Computational Science, Course content, Tools for teaching, Methods of instruction.

1. INTRODUCTION

In 2003 Wittenberg University created a Computational Science minor. This interdiciplinary minor, as with many Computational Science programs, lies at the intersection of mathematics, computer science, and the natural sciences (broadly defined). The goal is the application of computer technology to improve the understanding about the world around us. Indeed, much of the success of Wittenberg's Computational Science program has been in using computational software (such as Mathematica, Autodesk Maya 2008, Spartan '06, Excel, etc.) to construct and solve models in the natural and social sciences. Up to this point, all of the upper level electives for our minor had been housed in natural or social science departments.

The following course grew from a desire to provide a Computational Science elective in mathematics with the following properties.

- This course must have few prerequisites. We don't want to exclude students for lack of advanced mathematical experience.
- This course must not be a "black box". We want the student to truly understand the algorithms that are being implemented in a computational program and how they work.
- This course should address a modern technique. We want to show that new mathematics can be approachable to undergraduates, with the hope of exciting them about the field.
- This course should involve an in-depth *mathematical* study of a computational technique. We would want to develop the algorithm from first principles, prove theorems, and study consequences of the technique.
- This course must have theoretical (in addition to applied) connections to mathematics itself. In our computational science program, students rarely see how the computational skills, both symbolic and numeric, can advance abstract mathematics.
- This course should be platform independent. We don't want the student to learn a software package, but rather a process.
- This course absolutely must have extensive and meaningful applications to the sciences - hopefully to several distinct fields. After all, this is the essence of computational science.

A CAG course was chosen because it satisfies all of these criteria. For example, while CAG is an active area of research in both mathematics and computer science, it can be taught to any student with a prior introductory course in Linear Algebra, hence we can keep the prerequisites at a sophomore level. It satisfied the desire for a contemporary topic since CAG deals with an algorithm developed in the 1960's. The

algorithm is implemented on every major Computer Algebra System (CAS) so it is essentially platform independent.

But most importantly, this algorithm has major applications across the curriculum. At its heart, CAG deals with solving systems of polynomial equations, and as noted in the AMS review of [18], "A classic problem in mathematics is solving systems of polynomial equations in several unknowns. Today, polynomial models are ubiquitous and widely used across the sciences. They arise in robotics, coding theory, optimization, mathematical biology, computer vision, game theory, statistics, and numerous other areas." Indeed project topics in this course ranged across this spectrum.

This paper shows how the Computational Algebraic Geometry course was created. We begin with a quick overview of what CAG is. We then discuss the prerequisites and organization of the course. Next we examine the computational software that is utilized and describe the projects that students completed, one of which resulted in a publication for a student.

The rest of the paper concerns benefits and challenges of the course. It is our hope to show that this class can be a valuable elective for a Computational Science curriculum, and a useful module for a Computational Models or Algorithms class.

2. THE COMPUTATIONAL ALGEBRAIC GEOMETRY COURSE

2.1 Overview of Computational Algebraic Geometry

While algebraic geometry can be difficult and abstract, at its core it is merely concerned with solving systems of polynomial equations. The problem of simultaneously solving a system of polynomial equations is ubiquitous in mathematics. Undoubtedly our linear algebra students learned how to solve a system of *linear* equations using a method such as Gaussian Elimination or Cramer's Rule. In a more advanced class they may have learned some helpful techniques for solving a system of polynomials of arbitrary degree in *one* variable such as finding a greatest common divisor (gcd) using the Euclidean Algorithm or by computing a resultant.

These methods are classical, going back hundreds of years. However it wasn't until the 1960's that Bruno Buchberger came up with a generalization that took the "arbitrary number of variables" from Gaussian Elimination and the "arbitrary degree" from the Euclidean Algorithm and combined them into one powerful method called (predictably) Buchberger's Algorithm. Learning this algorithm was an essential part of the course.

The idea is that we start with a system of s polynomials in n variables with coefficients in a field K. We'll call it $\{f_1, \ldots, f_s\} \subseteq K[x_1, \ldots, x_n]$. By running Buchberger's algorithm one finds a different set of polynomials $\{g_1, \ldots, g_t\} \subseteq K[x_1, \ldots, x_n]$ with many nice properties. Two of the most essential are:

- The new set of polynomials must have an identical set of common zeros as the original polynomials (more specifically, they generate the same ideals in $K[x_1, \ldots, x_n]$).
- The new set of polynomials should (hopefully) be easier to solve than the original.

This new set of polynomials is called (not so predictably) a Groebner Basis for $\{f_1, \ldots, f_s\}$ after Buchberger's advisor, Wolfgang Gröbner (we've anglicized the spelling).

As implied above, a Groebner Basis can be defined over any field K, and indeed in this class we mathematically defined a field and gave many examples of them. However, in practice the course only dealt with when K was $\mathbb{R}$ or $\mathbb{C}$. The $\mathbb{R}$ case was useful when we were graphing examples and we would use $\mathbb{C}$ when we needed to completely factor our polynomials.

As a simple example, suppose that you wanted to find the intersection of a circle ($x^2 + y^2 - \frac{1}{4} = 0$) and a figure eight ($(x^2 + y^2)^2 - x^2 + y^2 = 0$) which comes down to simultaneously solving both of these equations. It is tedious to substitute and solve (and you could imagine that more complicated examples make this difficult.) However, if we run Buchberger's algorithm, we find a Groebner basis for the above system is $\{32y^2 - 3 = 0, 32x^2 - 5 = 0\}$ and from this we can immediately find the four intersecting points are $(\pm\sqrt{\frac{5}{32}}, \pm\sqrt{\frac{3}{32}})$. (See Figure 1). For details on the actual algorithm, please see the excellent text [4].

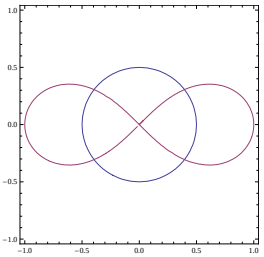


Figure 1: The Four Points of Intersection

Pertinent to this paper is the fact that this method is computationally intensive, though the algorithm itself is simple. All the major CAS such as Mathematica and Maple (and many others) contain Groebner Basis solvers, and several free software packages such as SAGE, Macaulay2, GroebnerFan are available on every platform. Wolfram's Alpha web site will compute Groebner Bases via the internet. This particular course utilized Mathematica, though some students downloaded SAGE or GroebnerFan onto their personal computers.

In 2008, Bruno Buchberger won the Association for Computing Machinary Paris Kanellakis Theory and Practice Award, which is awarded for theoretical advances in computer science that significantly affect the field. The award announcement states, "ACM (the Association for Computing Machinery) has recognized Bruno Buchberger, a professor at Johannes Kepler University in Linz, Austria, for his role in developing the theory of Groebner Bases, which has become a crucial building block to computer algebra, and is widely

used in science, engineering, and computer science. Buchberger's work has resulted in automated problem-solving tools to address challenges in robotics, computer-aided design, systems design, and modeling biological systems". [2].

This announcement proved to the students that this was a truly important method that had "significantly" changed computer science as well as multiple other fields. It also showed this was modern mathematics as compared to some of the classical techniques they are familiar with from Computational Science (such as Euler's method for Differential Equations or Lagrange Multipliers for Optimization).

2.2 Background of Students

Wittenberg University is a school of approximately 2000 undergraduate students with no graduate program in the sciences. We graduate approximately 14 math majors a year, 4 computer science majors, and 7 computational science minors. A similar course to ours is taught at many schools, but typically geared solely to mathematics majors, and often requires a modern algebra course and / or a course in programming. However, in an attempt to attract a broad audience to the Wittenberg course, the only prerequisite was a sophomore level linear algebra class. This was chosen since Buchberger's Algorithm is a generalization of some linear algebra algorithms presented in that class and a passing knowledge of linear combinations and independence is helpful. No knowledge of programming, Mathematica, proofs, etc. was required. This version of the course was an elective for both the math major and the computational science minor, hence fewer prerequisites. Thirteen students enrolled in the class, six of which were women. One faculty member sat in regularly.

2.3 Course Process

First offered in the spring of 2008, The course covered the first three chapters of the highly-recommended text *Ideals, Varieties, and Algorithms* by Cox, Little, and O'Shea [4]. Topics covered included:

- Basic definitions, such as ideal, variety, parametrization, generators, etc.
- Basic ideal theory.
- A review of polynomials in one variable and the GCD of a set of polynomials.
- Monomial orderings and monomial ideals.
- A division algorithm in $K[x_1, \ldots, x_n]$ (i.e. a division algorithm for several polynomials with each in multiple variables).
- Hilbert's Basis Theorem, Groebner Basis, Buchberger's Algorithm.
- Applications - Ideal membership, solving equations, elimination of variables, singular points, etc.

The first nine weeks of the semester consisted of three, hour-long lectures a week. In the classroom, there was one instructor computer with a projector, but no student computers. Mathematica was illustrated in the daily lectures but at this point much of the material lent itself to standard blackboard lectures. Relevant Mathematica examples were posted on the web. Weekly assignments, all which required Mathematica for either numerical calculations (usually for graphing) or symbolic computation (usually for polynomial manipulation), were collected and graded. Most computers on campus have Mathematica installed, and our site-license allows for students to install copies on their personal computers.

After we had covered Buchberger's Algorithm, the students chose project topics. The last six weeks of classes switched to two lectures a week and a one-hour lab which was held in a room with student computers so that students could work on their projects. There were two hour-long exams and a final. All exams consisted of an in-class portion (where all computation needed to be done by hand) as well as a take-home portion that required use of a computer.

2.4 Technology

While every calculation in the course could conceivably be done by hand, Mathematica was an essential part of the course. Not only did it save time and prevent algebra mistakes, it was also extremely valuable in plotting the curves and surfaces. Commonly used commands were:

- `Plot, ContourPlot, ContourPlot3D, Manipulate, ParametricPlot, ParametricPlot3D` for visualization.
- `PolynomialReduce, PolynomialGCD, PolynomialQuotientRemainder` for long division of polynomials.
- `GroebnerBasis` for the implementation of Buchberger's algorithm.

In Mathematica, computing the example of a circle and figure-eight is done in the following way:

$$\texttt{GroebnerBasis}[\{x^2+y^2-\frac{1}{4}, (x^2+y^2)^2-x^2+y^2\}, \{x, y\}]$$

which returns $\{-3+32y^2, -5+32x^2\}$.

The text has an Appendix on the GroebnerBasis implementations in AXIOM, CoCoA, Macaulay2, Magma, Maple, Mathematica, and SINGULAR so the text can be used with a wide variety of software packages.

2.5 Projects

Every student needed to complete an in-depth project which either looked at the applications to Groebner Basis to another field or examined a theoretical topic that we didn't cover in the course. Before choosing their topics, I took one class and discussed in generalities some possible projects, though it was clear that students were free to pick any topic they wanted. There were several intermediate deadlines designed to keep the projects on track to finish on time. Students worked in pairs, and had to create both a poster and a paper. The posters were presented at a class "open house" where other faculty and students attended. We did run into computational limitations with some of the projects, usually concerning memory. All students were able to complete at least modified versions of their stated projects. Since running the course in the spring of 2008, Wittenberg has placed

Mathematica on their computing cluster, which will be valuable in future iterations of the course.

Below is a short description of the projects from the course, as well as a (very) short bibliography for additional reading, hopefully showing the breadth of applications. Computation entered into these projects in a variety of ways, though typically the students created a model using polynomial equations, and then computed a Groebner Basis for that system. They had to "solve" the model by extracting the relevant information from that Groebner Basis.

2.5.1 Theoretical Projects

A Walk with Groebner A Groebner Basis is not unique, but rather depends on a choice called the "monomial ordering". It turns out that an interesting project is determining which "monomial orderings" give the same Groebner basis and which give distinct ones. This partitions the space of all monomial orderings into a so-called Groebner Fan.

The run-time of the algorithm depends heavily on this choice of ordering. It turns out that it may be faster to compute the Groebner Basis for a "fast" monomial ordering and convert that Groebner Basis to a second basis associated to a "slow" ordering than to just compute the basis with the "slow" ordering at the outset. The algorithm involves creating a path in the space of all monomial orderings that starts at the current ordering and ends at the desired ordering. When traveling along the path and you cross a wall in the Groebner Fan, the Groebner Basis will change. Keeping track of these changes will convert the original Groebner Basis to the new one. This algorithm for converting between bases is called a Groebner Walk, and is implemented in Mathematica and other packages. [8] [3] [6].

Solving the Frobenius Problem Using Groebner Bases This project dealt with a question in number theory called the Frobenius Problem. It asks if given a set of nonnegative numbers $\{a_1, a_2, \ldots, a_n\}$ with $gcd = 1$, then what is the largest nonnegative number that can't be written as a combination of these numbers with positive coefficients.

For example, consider the set $(4, 9)$. Since $gcd(4, 9) = 1$, any number can be written as a combination of 4 and 9 if we allow negative coefficients. Obviously small numbers such as $1, 2, 3, 5, 6, etc.$ can't be written as a combination with positive coefficients. The Frobenius problem asks what is the largest such number (and there always is one). In this case it is 23. The students that did this project were computer science students that implemented the algorithm themselves. They appreciated seeing how modern techniques were being used on problems from over 100 years ago. [7] [17].

2.5.2 Biological Applications

Reverse-Engineering Biochemical Networks using Gröbner Fans The project involved a paper by Laubenbacher and Stigler which uses Groebner bases to approximate the most likely dynamic model of regulatory biochemical networks. This project also required learning about Groebner Fans as it essentially ranked each of the cells in the fan, returning the cell with the greatest score. This cell corresponded to the most likely model. [10].

2.5.3 Chemical Applications

Molecular Modeling with Groebner Bases This pair of students dealt with determining potential configurations of rings of carbon atoms, if we assume that all the bond lengths and bond angles are the same between adjacent atoms. While there are multiple ways to model these configurations, we'll quickly describe the setup the students used for cyclopentane.

Imagine the five carbon atoms in $\mathbb{R}^3$, and assume that the bond lenghts between adjacent atoms is 1. By rotating the system, we can assume that the five carbon atoms have coordinates (read off in a clockwise direction):

$$(0, 0, 0), (l, m, 0), (x, y, z), (a, b, c), (1, 0, 0)$$

The fact that adjacent atoms are distance 1 away from each other is encoded by polynomials such as

$$(l)^2 + (m)^2 - 1 = 0, (x - l)^2 + (y - m)^2 + (z)^2 - 1 = 0, etc.$$

The students then forced the *angle* between any two bonds to be the same by introducing a new variable t and requiring the *distance* between two non-adjacent atoms to be t. It is clear that if the distance between any two carbons that have one carbon between them is the same, then the angles formed by any three carbons is also the same. This fact is encoded by equations such as

$$(x)^2 + (y)^2 + (z)^2 - t = 0, (a - l)^2 + (b - m)^2 + z^2 - t = 0, etc.$$

By computing a Groebner Basis for these equations, and finding all real solutions to that system, this pair of students was able to prove that any ring of five atoms must be planar. This technically isn't true for cyclopentane since hydrogen atoms force one angle to be a bit different from the rest. However, as a model it was very successful. After, cyclopentane, they moved onto cyclohexane and were able to isolate the "chair" and "boat" isomers for rings of six carbons. [11] [5].

2.5.4 Economics Applications

Nash Equilibrium and a 'Very Simple' Game of Poker This project worked through an example from Bernd Sturmfels text *Solving Systems of Polynomial Equations*. [18]. In the project, students used Groebner Bases to recover a 1950 game theoretic- result of John Nash concerning the Three Person Poker Game. The students calculated the Nash Equilibrium for the game, giving optimal strategies for the players. [12] [13].

This project was especially exciting for our Computational Science program as we are working hard to create connections between computation and some of the social sciences such as economics. Showing how Groebner Bases can interact with Game Theory and Algebraic Statistics ([14] [16]) may open many new interdiciplinary connections.

2.5.5 Recreational Applications

Solving N-Colorable Graphs with Groebner Bases While this may seem like a pure math topic, a wide variety of puzzles can be rephrased in terms of graph colorings, and those types of problems can be solved using Groebner Bases. This group solved problems such as Sudokus, Magic Squares, Latin Squares, Kakuro puzzles, etc.

We can see how the model is created from a very simple 3×3 Latin Square. This is a 3×3 grid, which we want to fill in with 1's, 2's, and 3's so that no number is repeated in a row or column. We start by labeling our entries with variables.

a	b	c
d	e	f
g	h	i

Our model will consist of two types of equations. *Domain* equations encode the fact that every entry must be $1, 2$ or 3. They look like

$$(a-1)(a-2)(a-3) = 0,\ (b-1)(b-2)(b-3) = 0,\ etc.$$

It is clear that these polynomials will vanish exactly when the variables are in the desired domain. *Distinctness* equations encode that no entry may be repeated. These are encoded by introducing a "dummy" variable (we use x, y and z here). The equations that force the first row to all be distinct look like

$$(a-b)x = 1,\ \ (b-c)y = 1,\ \ (a-c)z = 1$$

This is done for every row and column. Notice that these equations will only have solutions if the entries are unequal. Otherwise, we get an equation of $0 = 1$.

After creating a system of all these equations, a Groebner Basis is calculated from it. At this point, there are techniques to count the number of solutions to the system, which in turn gives the number of 3×3 latin squares [3] [1].

Marshall Zarecky, one of the members of this group, used these techniques and published a paper in the Proceedings of the Midstates Conference on Undergraduate Research in Computer Science and Mathematics (2008). He used Groebner Bases to find all the solutions to an old Milton Bradley game "Drive Ya Nuts" and solved Cipra's Puzzle. [19]. Another student that was not in the class but worked on a summer reserach project with me, published a paper after writing a Mathematical Program that used Groebner Bases to solve Ken Ken Puzzles. [9].

3. BENEFITS AND CHALLENGES

Free response questionnaires were given after each test. Upon the completion of the course, two exit evaluations were also administered. One was a Quantitative Course Evaluation sheet, while the other was an open response writing (qualitative) evaluation. What follows is taken from these end of the year evaluations. This isn't intended as a scientific assessment of the learning in the course, but rather as a metric of student experience.

3.1 Challenges

Since the prerequisites were set at such a low level, there was a wide variety of abilities in the class. The course had Seniors, Juniors, Sophomores and one advanced High school student (not to mention the one mathematics faculty member who was auditing). This can be a huge challenge for any class. However, since few students had even heard of an ideal, and no student had ever heard of a Groebner Basis or Buchberger's Algorithm, this mitigated much of the difference in mathematical experience.

There were also large differences in the interests of the students (as illustrated by the wide variety of projects). It could have been a problem to try to satisfy the chemists, the educators, the computer science majors, and mathematicians that were in class. Luckily, this technique has such broad applications, everyone was able to find a project that pertained to their interests.

Almost universally, people felt that this class was hard and required a lot of work, which certainly is true. Of the 11 qualitative responses $n = 9$ students commented the course was hard. This was also see on the 13 quantitative responses where $n = 12$ said that the material was either "more" or "much more" difficult than other courses and $n = 11$ said they worked harder on this course than their others. I agree that the course was challenging and therefore I needed to overcome some frustrations on the part of the students.

3.2 Benefits

It may appear with so many students finding the course difficult, that they would dislike it. Quite the opposite was true. While there were two students that appeared to be dissatisfied by the course, by far most students ranked it as an excellent course and enjoyed the material. On the 13 quantitative responses $n = 9$ ranked the course at excellent, $n = 10$ thought the course "demonstrated the importance and significance of the subject matter", and $n = 13$ responded that the course frequently "introduced stimulating ideas about the subject."

On the qualitative responses, students enjoyed learning "new" and "modern" mathematics ($n = 5$) and found the projects valuable ($n = 3$). All students that filled out a qualitative evaluation would recommend the course to their peers ($n = 11$).

Often "computation" in the context of Computational Science refers to numerical methods. Certainly at Wittenberg, students would have much exposure to numerical computational techniques. This class had the benefit of including both numerical and symbolic computation. This may have been the students' first experience with symbolic computation as it related to computational science.

3.2.1 Module For General Computational Science Students

After the success of this course, the author created a Mathematica notebook to serve as a short module for our Computational Models and Methods class. As mentioned above, this course serves as our Introduction to Computational Science. Obviously the module didn't go into the depth that the course did, but it did explain how the algorithm can be used to solve many of the mathematical models that are already studied in that course. In the module, Groebner Bases are used to find max/mins using Lagrange Multipliers, solve equilibrium solutions of linear differential equations, project onto a subspace for computer graphics, and solve linear programming problems, which are all topics covered elsewhere in the course. This notebook, entitled "A Groebner Basis Module for Comp 260" can be viewed at CSERD at [15].

We feel that this is a particularly important algorithm to

highlight since many automated commands in computational software (for example `Solve` and `NSolve` in Mathematica) utilize Buchberger's algorithm when called. Showing them how to use Groebner Bases explicitly makes the program less of a "black box". This module is currently being used in the course.

4. CONCLUSIONS

An undergraduate course in Computational Algebraic Geometry, while novel and somewhat challenging, has many benefits.

At the end of the course, each student had the ability to:

- algebraically find the common solutions to an arbitrary number of polynomials in an arbitrary number of degrees $\{f_1, \ldots, f_r\}$.
- determine if another polynomial g could be written as a combination of those f_i.
- understanding how these algebraic relationships affected the corresponding plots and geometry. In particular they could determine equations for the union, intersection, and projection of geometric objects given by the zeros of polynomials. They could also find singular points of these objects.
- use these techniques to solve optimization, linear programming, equilibrium, etc. problems.
- construct polynomial mathematical models of a variety of types, and solve them using this technique.

In short, it allows for an in depth study of an extremely useful algorithm with applications to almost every natural science. It permits students to walk the line between theoretical mathematics and computational science and see how they each benefited the other. Most importantly, it has an extremely wide variety of applications which students find very appealing.

5. ACKNOWLEDGEMENTS

This work was partially supported by the federal grant for enhancement of computational science at Wittenberg University and by Wittenberg University itself. I would like to thank Prof. Al Stickney for his helpful suggestions in teaching this course. In addition, I thank Prof. Emeritus Jim Noyes, Dr. Ray Dudek, and the referees for their suggestions that improved this manuscript.

6. REFERENCES

[1] Adams, W. Loustaunau, P.: An introduction to Gröbner Bases. Graduate Studies in Mathematics. 3, Amer.Math.Soc. Providence, RI (1994)

[2] AScribe Newswire: ACM Honors Innovator of Automated Tools for Mathematics; Bruno Buchberger Developed Algorithm Used in Computer Algebra to Solve Problems in Computer Science, Engineering, Science. `http://www.ascribe.org/cgi-bin/behold.pl?ascribeid=20080513.091858&time=11%252` (May 13, 2008).

[3] Cox, D.: A Groebner Basis Tutorial. `http://www.cs.amherst.edu/~dac/lectures/gb2.handout.pdf` 31-50 (2007)

[4] Cox, D., Little, J. O'Shea, D.: Ideals, Varieties, and Algorithms - An Introduction to Computational Algebraic Geometry and Commutative Algebra. Springer (2000)

[5] Emiris, I.Z. Mourrain, B.: Computer Algebra Methods for Studying and Computing Molecular Conformations. Algorithmica. 25 , 372-402 (1999)

[6] Evans, G.A.: Noncommutative Involutive Basis. Ph.D. Thesis - University of Wales. `http://arxiv.org/PS_cache/math/pdf/0602/0602140v1.pdf`, (2005)

[7] Frobby - A software package for computing Frobenius numbers and irreducible decompostions of monomial ideals. `http://www.broune.com/frobby` (2006)

[8] Fukada, K. Jensen, A. N., Lauritzen, N. Thomas, R.: The Generic Gröbner walk. J. Symbolic Comput. 42, 298-312 (2007)

[9] Griffith, A. Parker, A.: A Groebner Basis Approach to Number Puzzles. In: Proceedings of the Sixth Annual MCURCSM Conference 2009, pp 57-64 (2009)

[10] Laubenbacher,R. Stigler, B.: A computational algebra approach to the reverse engineering of gene regulatory networks. J. Theoret. Biol. 229, 523-537 (2004)

[11] Michelucci, D. Foufou, S.: Using Cayley-Menger Determinants for Geometric Constraint Solving. In: ACM Symposium on Solid Modeling and Application, pp. 285-290 (2004)

[12] Nash, J.:, Non-cooperative games. Annals of Math. 54, 286-295 (1951)

[13] Nash, J. and Shapley, L.: A simple three-person poker game. In: Contributions to the Theory of Games, pp. 105-115. Princeton University Press, Princeton, NJ (1950)

[14] Pachter, L. and Sturmfels, B.: Algebraic Statistics For Computational Biology. Cambridge University Press. New York (2005)

[15] Parker, A.: A Groebner Basis Module for Comp 260". The Computational Science Education Reference Desk (CSERD). `http://www.shodor.org/refdesk/Catalog/`, (2010)

[16] Pistone, G., Ticcomagno, E., and Wynn, H.P.: Algebraic Statistics - Computational Commutative Algebra in Statistics. Monographs on Statistics and Applied Probability. 89, Chapman & Hall (2001)

[17] Roune, B.H.: Solving Thousand Digit Frobenius Problems Using Groebner Bases. J. Symbolic Comput. 43, 1-7 (2008)

[18] Sturmfels, B.: Solving Systems of Polynomial Equations. CBMS Regional Conferences Series. 97, Amer.Math.Soc. Providence, RI (2002)

[19] Zarecky, M. Parker, A.: Describing A Combinatorics Problem with a System of Polynomial Equations. In: Proceedings of the Fifth Annual MCURCSM Conference 2008, pp. 101-109 (2008)

Using WebMO to Investigate Fluorescence in the Ingredients of Energy Drinks

Mark Smith
Arthur CUSD #305
301 E. Columbia St.
Arthur, IL 61911
001-217-543-2146
msmith@arthur.k12.il.us

Emily Chrisman
Pleasant Plains CUSD #8
500 N. Cartwright St.
Pleasant Plains, IL
001-217-626-1044
echrisman@ppcusd8.org

Patty Page
PBL CUSD #10
600 Orleans St.
Paxton, IL 60957
001-217- 379-3314
pageps@earthlink.net

Kendra Carroll
Shiloh CUSD #1
21751N 575th St.
Hume, IL 61932
001-217-887-2365
kcarroll@ista-il.org

ABSTRACT
With computers becoming more powerful tools, computational modeling can be introduced gradually to secondary students allowing them to visualize complex topics and gather data in the different scientific fields. In this study, students from four rural high schools used computational tools to investigate attributes of the ingredients that might cause fluorescence in energy drinks. In the activity, students used the computational tools of WebMO to model several ingredients in energy drinks and gather data on them, such as molecular geometry and ultraviolet-visible absorption spectra (UV-Vis spectra). Using the data they collected, students analyzed and compared their ingredient molecules and then compared them to molecules that are known to fluoresce to determine any patterns. After students participated in this activity, data from testing suggest they were more aware of fluorescence, but not more aware of how to read an UV-Vis spectrum.

Categories and Subject Descriptors
K. Computing Mileux, K..3 Computer and Education, K..3.1 Computers uses in education

General Terms
Measurement, Performance, Design, Human factor, Experimentation, Theory

Keywords
Fluorescence, Molecular Geometry, WebMO, Molecular Shape, Energy Drink, Computational Science, Education, Chemistry, High School, Absorption Spectra, UV/Visible

1. INTRODUCTION
Students today have changed greatly in the methods that they use to seek and obtain information in their lives. Teachers who are aware of this change can adapt their methods to better serve their students. These adaptations should enhance students' educational experience and increase their comprehension of the subjects being taught. An important difference between students today and those in the past is their use of desktop, laptop, netbook, and tablet personal computers (PCs). They have become well versed in the use of computers for information gathering and entertainment. These computers are capable of running programs that can create highly accurate simulations and models in chemistry. In an effort to determine the impact of computer simulations and computational models on student comprehension in chemistry, the National Science Foundation (NSF) is funding the Institute for Chemistry Literacy through Computational Science (ICLCS), a five year program hosted by the University of Illinois at Urbana-Champaign (S.Sendlinger, personal communication, ICLCS workshop 2008). The ICLCS program has given rural chemistry teachers from all over the state of Illinois, including over ninety school districts, the opportunity to come together and be introduced to some computational tools and trained in their use. After the training, these teachers were asked to create an "Ice Cube," which is a small unit that could take one to two class periods to complete and uses computational tools. The teachers were asked to test their "Ice Cubes" in their classes and gauge the effectiveness by a pre/post test system as well as qualitative observations. The use of computational tools in the classroom is believed to help students understand some of the more abstract and hard to visualize concepts in chemistry. It does this by giving students the ability to manipulate visual models of the conceptual aspects of chemistry. Manipulations in these computational tools allow students to "see" and experiment with molecules and their behavior in a more concrete way.

The "Ice Cube" used in this study used computational tools to introduce fluorescence to the students. It incorporated real-world scenarios, inquiry, research, complicated modeling and molecular computations to further the student's knowledge of this material. Students were shown the fluorescence of a common energy drink. This started the inquiry process of why this happens. The students researched the ingredients and fluorescence to further their

understanding. The teachers in leading this project introduced the computational tool WebMO to help the students further their investigation.

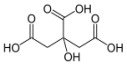
2. METHODS

2.1 Instructional Design

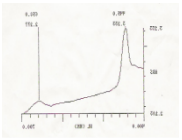
This particular "Ice Cube" was envisioned as a part B in a three part plan where A is introductory material such as Molecular Shape, Bonding, or Orbitals/Energy levels, etc. Part C would be a springboard to transition into another area of study using information or practices used in the "Ice Cube". In this study, to gain the attention of the students, they were given the illustration of common energy drinks glowing under a black light. This illustration was designed to start the students on an inquiry investigation on the topic of fluorescence. The students were encouraged to collect data and hypothesize why some molecules fluoresce. They were guided using positive questioning techniques to explore the molecular geometry and the ultraviolet visible absorption spectra to collect data that may lead to an explanation of what causes this phenomenon. In each classroom, students were split into four teams and investigated two different common ingredients including citric acid, vitamin B6, vitamin C, caffeine, vitamin B2, glucose, fructose, and vitamin A. Students used WebMO (http://webmo.ncsa.uiuc.edu), a computational program that can be used to calculate molecular geometries and ultraviolet visible absorption spectra, in addition to many other parameters. The program uses a powerful set of computations run on supercomputer-based servers to calculate the geometry, spectra, bond energies, and other data after the user inputs the basic shape of a molecule. WebMO's optimized geometry and computed UV-Vis spectra of the common ingredients of energy drinks were analyzed by students to see if they could find any indications that the material might fluoresce.

The total sample of ninety students had similar experiences in three of the classrooms, with 24 of the students being in the control group. The Chemistry I students were mainly in tenth grade, a few students were in the eleventh and twelfth grades. The Anatomy and Physiology class mainly consisted of eleventh and twelfth grade students. These students were first administered the pretest with no instruction. The pretest included questions on fluorescence, basic chemical symbols, graph reading, and emission spectra. The questions on the pretest and post test were:

1. Define fluorescence
2. What causes fluorescence?
3. What does the double line in the figure represent?



   a. double strength bond
   b. double bond
   c. two carbon atoms present
   d. more hydrogen present



4. Does the emission spectrum above show visible light?
   a. Yes
   b. No
   c. Not enough information
5. Using the same graph, what energy value has the greatest intensity?
   d. a. 658.nm
   e. b. 445 nm
   f. c. 3.233
   g. d. 2.227

The students then completed the "Ice Cube." Students were broken into small groups of two or three and subsequently into four different teams. The students were given the Student Instruction sheet and Student Datasheet. Once students were broken into teams, they used WebMO to build an ingredient molecule from the energy drink using a provided structural formula, as in **Figure 1.**

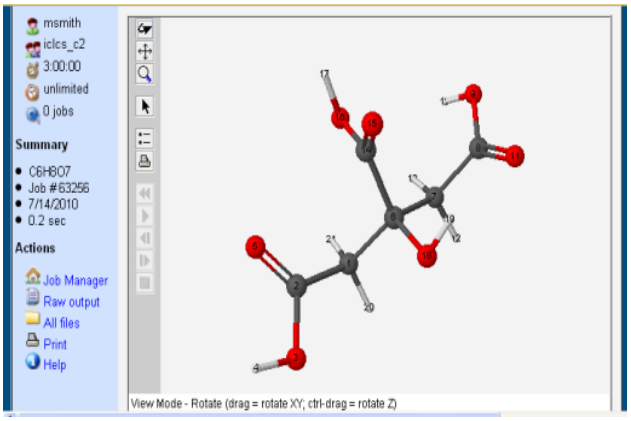


Figure 1 Citric acid built in WebMO

Students recorded the UV-Vis spectrum and structure in their data sheet. Then students imported a completed, more complex ingredient molecule into WebMO from the National Institute for Science and Technology's Chemistry WebBook website (http://webbook.nist.gov/chemistry/) (2008). The students then performed the same processes and recorded their results on the data sheet, as in **Figure 2 and Figure 3.**

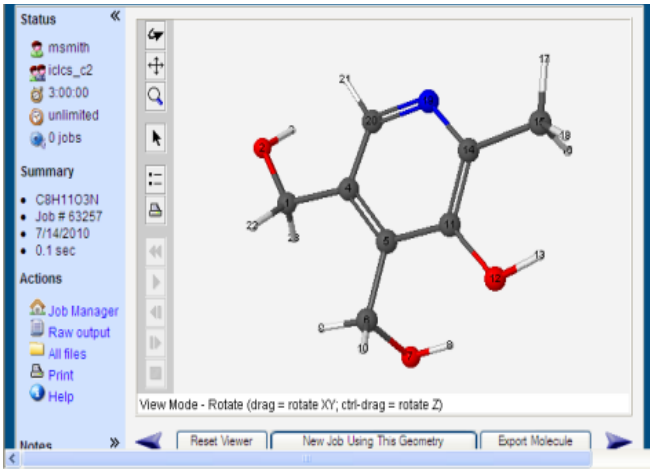


Figure 2 Vitamin B6 imported and optimized in WebMO

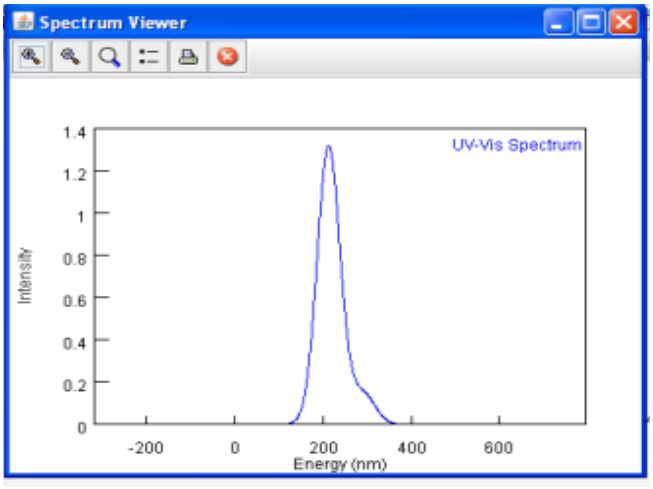


Figure 3 UV-Vis Spectrum of Vitamin B6

Following these activities, students answered a series of data analysis questions to try to determine if there were similarities between the two ingredients. The students then compared their data to the data of chemicals that were known to fluoresce and commented on whether or not they thought their ingredients would fluoresce. Finally, the class discussed the possibility of any one of these ingredients being the one that caused the fluorescence in the energy drink. The class also discussed why literature on the cause of fluorescence in energy drinks seemed to be nonexistent. After the "Ice Cube", thirty-one Chemistry I students and ten Anatomy and Physiology students received traditional instruction on fluorescence and the emission spectra. The other twenty-five students participating in this "Ice Cube" did not receive traditional instruction. In the end of the lesson, all students took the same assessment that was given in the beginning of the "Ice Cube" as the post test. The post test responses indicate that students may have greater understanding in these topics.

2.2 Computational Tools

After creating the model of the molecule using the editor in WebMO (http://webmo.ncsa.uius.edu), students first ran the molecule through a geometry optimization using MOPAC/PM3 modeling parameters. This step optimized the model into its lowest energy state. The students then found the ultraviolet visible absorption spectrum using the Hartree-Fock/Other (null) modeling parameters, as in Figure 4.

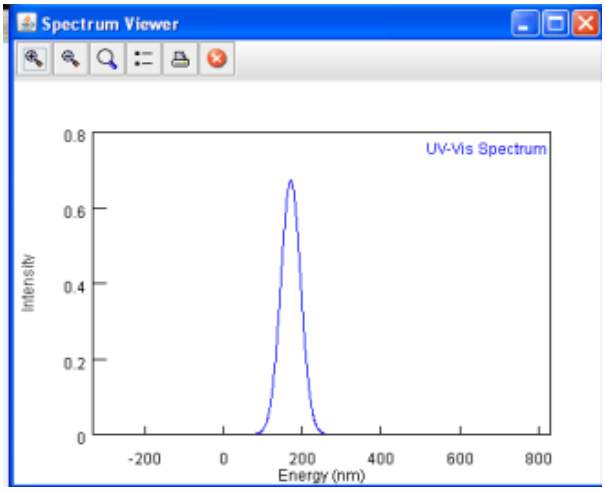


Figure 4 UV-Vis Spectrum for Citric Acid

3. RESULTS

The following results, in **Table 1** were gained by having sixty-six students in three high school Chemistry I classes and one high school Anatomy and Physiology class participate in this activity. These classes were in three different rural Illinois schools. Of the sixty-six students, all of them took a pretest and post test for the activity and had the following results. The table also compares Chemistry I students who did receive traditional instruction after the WebMO activity versus the Chemistry I students who did not. Twenty-four Chemistry I students acted as a control and were given the pretest and post test after traditional instruction only. The data in **Table 1** shows the overall improvements gained by each type of student.

The control group, receiving only traditional instruction, had a gain of 0.65 points. The Anatomy and Physiology group showed a gain of 1.7 points after using WebMO. The Chemistry I students with both WebMO ice cube and traditional instruction showed a gain of 0.55 points and the Chemistry I students with just the ice cube showed a gain of 1.12 points. The students with the WebMO experience also always had a higher post test score than the control group. This data shows that significant gains were achieved with the use of WebMO. When compared to the control group with traditional instruction only, it was found that the use of WebMO, on average, did have the most gain of understanding on the topic of fluorescence. It was also noted that the free response answers on the pretest were more indepth for the students who participated in the project and did some research on their own. The students receiving only traditional instruction had answers ranging from "no clue" on the pretest to "emission of a

photon" when asked to define fluorescence. Although the final answer is a component of the correct answer, it was not a complete answer.

Table 1 Pre/Post Test Results

Classes	Pretest average score out of 5	Posttest average test score out of 5
Anatomy and Physiology 10 students	1.8	3.5
Chemistry I students with traditional instruction 31 students (2 classes)	2.85	3.4
Chemistry I students without traditional instruction 25 students	2.08	3.2
Chemistry I students with only traditional instruction 24 students	1.25	1.9

All students in the classes tested made improvements from their pretest to the post test scores, which may indicate an increased understanding of what fluorescence is and how it works. Students' answers on the short answer part of the assessment became more in depth and comprehensive. On the pretest, examples of their answers to the question "Define fluorescence." were; blank answers, "a molecular process", "the amount of energy to complete something", "chemical substances made from fluorine", and "I don't know". Of the twenty-five Chemistry I students, twenty-three students were unable to properly define fluorescence. Most students seemed to have no prior knowledge of what fluorescence was before the activity. The activity seemed to improve their understanding of what fluorescence is and what causes compounds to fluoresce. On the posttest, answers to the same question become more in-depth, like "the emission of light after an excited state".

4. DISCUSSION

Research has shown that fluorescence is best understood by looking deeper than just the geometry of a molecule. Although double bonds may be a good reason to look deeper, the possible transitions that cause fluorescence are effected by not only the arrangement of nonbonding and bonding orbitals within the molecule but also the environment such as hydrogen bonding. Moreover because energies are effected by various conditions including the orientation of the p orbitals, the circumstances leading to fluorescence in molecules differ depending on whether the molecule is azoaromatic, or a carbonyl, etc (Sidman, 1958). Because understanding the reasons for fluorescence are so complex students would not be able to reach any clear answer, but should be able to hypothesize some generalities.

Students were able to perform the energy drink activity easily. Most of the students had previous working knowledge of WebMO and the team structure allowed the inexperienced students to complete the lab. The directions were detailed and step-by-step preventing students from making many mistakes. Unfortunately, there was a problem in the procedure for importing molecules that was addressed in the student instructions. The student instructions failed to tell students after importing their second molecule to open the editor in WebMO and select "Cleanup; Comprehensive mechanics" to attach hydrogens where needed. This mistake caused numerous failures of their model runs for several groups, which began to frustrate some of the students until the mistake was found and corrected in the instructions. Students also had trouble building the molecules they were assigned. However, with teacher guidance they were able to rebuild the molecule quickly and complete the lab without incident. It was also observed that once one student was shown how to overcome a technical problem in WebMO, they would help others around them. This helped foster cooperative learning. Overall, the instructions were well written and allowed the activity to be completed in a timely manner.

The data sheet also had its pros and cons. Overall, the data sheet was easy for students to follow and complete. Some students needed clarification with a few of the questions; however, the data sheet was clear and concise. Unfortunately, there was a problem with student understanding in the questions that dealt with the ultraviolet visible absorption graphs. Students consistently missed questions dealing with the ultraviolet visible absorption graphs in the WebMO output and how to interpret them as evidenced in the post test. More time discussing these graphs may have furthered understanding. Also, some students answered some questions incompletely, listing only one similarity between substances. Overall, students were able to perform the activity in an acceptable time with minimal questions. The post test seems to indicate that this experience did allow students to gain knowledge of fluorescence and factors that could perhaps cause it.

In the chemistry 1 class during the post test, out of the twenty-five students, sixteen consistently missed at least one of the two questions that referred to the emission spectrum graph. Even after completing the lab, they were unable to decide if the wavelength or the intensity indicated the visibility of light. They also were unable to distinguish which part of the graph, the intensity or the wavelength, indicated the amount of energy being released. Also, in the laboratory data sheets students only recognized one similarity in either the molecular geometry or the ultraviolet visible absorption spectrum. They were unable to find multiple similarities or find similarities in both the molecular geometry and ultraviolet visible absorption spectrum. Finally, students were unable to draw the conclusion that using only ultraviolet visible absorption and molecular geometry did not give them enough information to determine if a molecule would fluoresce or not. The students tested did not understand that this is a current area of study and a definitive answer is not available. After the post test, the control group of twenty-three students had nine students correctly identify if light was visible on an emission spectra graph. None, however, could identify which part of the graph, the

intensity or the wavelength, indicated the amount of energy being released. This showed that the project did allow for further understanding of the energy associated with the emission spectra.

To fix the problems faced in this activity some changes should be implemented. The first clear change would be to add steps in the instructions for importing molecules. We need to clarify that the students should clean-up their molecule to add hydrogen in the needed places for the proper computation to take place. The data from the students supports the idea that this "Ice Cube" is best used as a part B where A is the introductory material for the subject being studied that could include listing and identifying molecular substructures such as carbonyl, nitroso, azo, and other hetero groups as well as aza aromatic compounds or simply recognizing ring structures, double-bonds and alternating bonds. Other introductory material could include experience with optics, the electromagnetic spectrum, light intensity, emissions, wavelength and frequency. Using the data from the students, the activity should be edited to help achieve more understanding of ultraviolet visible absorption and the differences in molecular geometry (J. Sidman, 1958). There should be more introduction and discussion of the ultraviolet visible absorption, and what wavelength can be represented and the visible spectrum. Finally, students should be prompted in the data sheet to look more deeply into similarities in geometry and ultraviolet visible absorption. This activity was designed to illustrate the unknown properties of scientific discovery. Students have very little contact with the unknown in the traditional classroom setup. Very little can be done to improve the frustration of the students in an activity where the outcome or answer is unknown. This can only be eased with more exposure to this type of activity.

Ultimately this activity allowed for the students to be active participants in the scientific method and in learning the complexity of determining a cause for fluorescence. The students were asked to gather information, ask questions, submit a hypothesis, test this hypothesis and discuss the results. They were put into a situation where they were testing a problem without a clear answer. This confused most students but also gave experience in real-world science. The students gained insight on doing research when there is no clear answer. This model, when compared to the control group, showed furthered understanding of this topic and scientific reasoning.

5. CONCLUSION

WebMO allowed students to build complex virtual molecules that they would have difficulty building in the classroom using other modeling techniques, such as ball and stick models. Also, WebMO allowed students to use the molecule they built to find new information about each molecule. Students looked at structure, the molecular geometry as well as ultraviolet/visible absorption in hopes of understanding how these features might affect its fluorescence. From the data collected, using WebMO allowed students to gain a better understanding of the concept of fluorescence. The experiment also allowed students to participate in a lab that would otherwise require some very expensive equipment to find the spectra of the molecules or require the use of literature to find the spectra. The "Ice Cube" allowed them to be active participants in the learning of the scientific method and how complex the topic of fluorescence is. WebMO allowed students to visualize the molecules they studied instead of just looking them up in reference books. WebMO is a powerful tool that allows students to investigate electron density, infrared spectra, NMR and other data. With today's students being more and more technologically savvy, WebMO can be used to help introduce complex topics that are almost impossible to visualize in real life.

Ultimately, this activity did succeed in encouraging and fostering scientific inquiry in the students. The students gained experience in taking real-world occurrences and applying the scientific method to understand the mechanisms behind observations. This activity also allowed the students to apply learned topics in chemistry to their everyday experiences. This activity did improve student understanding. Students, when compared to the control group, had a more in-depth knowledge of fluorescence, molecular structure, graph reading after participating in this activity. Further improvements will make this an even more successful way of introducing these topics.

6. ACKNOWLEDGEMENTS

We express our sincere appreciation to the ICLCS staff and members for their support and encouragement on this paper and the development of this "Ice Cube". We also would like to thank the University of Illinois for the use of their facilities and supercomputers and NSF for their funding of this program.

7. REFERENCES

[1] NIST Chemistry WebBook (2008). *Molecules used for importing*. Retrieved from: http://webbook.nist.gov/chemistry/

[2] Sidman, Jerome W.; Electronic Transitions Due to Nonbonding Electrons in Carbonyl, Aza-aromatic, and Other Compounds; Dept. of Chemistry, Cornell University, Ithaca, New York March 1, 1958

The Use of Spreadsheets and Service-Learning Projects in Mathematics Courses

Morteza Shafii-Mousavi
Indiana University South Bend
P.O. Box 7111
South Bend, IN 46634
574-520-4516
mshafii@iusb.edu

Paul Kochanowski
Indiana University South Bend
P.O. Box 7111
South Bend, IN 46634
574-520-4516
pkochano@iusb.edu

ABSTRACT

In the Indiana University system, as well as many other schools, finite mathematics is a prerequisite for most majors, especially business, public administration, social sciences, and some life science areas. Statisticians Moore, Peck, and Rossman [23] articulate a set of goals for mathematics prerequisites: including instilling an appreciation of the power of technology and developing skills necessary to use appropriate technology to solve problems, developing understanding, and exploring concepts. This paper describes the use of Excel spreadsheets in the teaching and learning of finite mathematics concepts in the linked courses Mathematics in Action: Social and Industrial Problems and Introduction to Computing taught for business, liberal arts, science, nursing, education, and public administration students. The goal of the linked courses is to encourage an appreciation of mathematics and promote writing as students see an immediate use for quantitative and communication skills in completing actual service-learning projects. The courses emphasize learning and writing about mathematics and the practice of computer technology applications through completion of actual industrial group projects.[1] Through demonstration of mathematical concepts using Excel spreadsheets, we stress synergies between mathematics, technology, and real-world applications. These synergies emphasize the learning goals such as quantitative skill development, analytical and critical thinking, information technology and technological issues, innovative and creative reasoning, and writing across the curriculum.

Keywords

Excel spreadsheet demonstrations, service-learning Projects, linked mathematics and technology courses

[1]A detailed discussion of various aspects of these courses is found in [13],[14],[15],[16],[17],[34],[35],[36]

1. INTRODUCTION

Business and economics students generally are required to take introductory mathematics courses, traditional finite mathematics and calculus, as prerequisites for statistics, operations research, finance, and other quantitative business and economics offerings. Such prerequisites often are offered in the mathematics department and taught with little regard to how students will use the concepts and techniques in their majors. O'Shea and Pollatesek [25], for example, argue that the traditional curricular structures from elementary through junior year college mathematics courses (algebra, geometry, pre-calculus, and calculus) do not allow students to encounter the range of ideas embraced by modern mathematics. Students seldom see these ideas used and rarely have sufficient time for real mastery. Statisticians Moore, Peck, and Rossman [23] believe the highest priority requirements of statistics from the mathematics curriculum to be that students develop skills and habits of mind for problem solving and for generalization.

Along these lines Moore, Peck, and Rossman [23]articulate a set of goals for mathematics prerequisites:

1. Emphasize multiple presentations of mathematical objects.
2. Provide multiple approaches to problem solving, including graphical, numerical, analytical, and verbal.
3. Adopt learning-centered instruction and address students' different learning styles by employing multiple pedagogies.
4. Insist that students communicate in writing and learn to read algebra for meaning.
5. Use real, engaging applications through which students can learn to draw connections between the language of mathematics and the context of the application.
6. Instill appreciation of the power of technology and develop skills necessary to use appropriate technology to solve problems, develop understanding, and explore concepts.
7. Align assessment strategies with instructional goals.

Moore, Peck, and Rossman (hereafter MPR)[23] note that statisticians are less concerned with specific content coverage but more with the development of the skills and habits of

mind to solve problems, generalize ideas, and model service-learning problems. In short, satisfying these goals necessitates designing a freshman-level math course that exposes students to a variety of mathematical ideas and technological applications that prepare students to deal with statistical topics, data, and modeling.

2. INNOVATIVELY LINKED MATHEMATICS AND TECHNOLOGY COURSES

Mathematics in Action: Social and Industrial Problem is a project-based mathematics course, [15] and [36], which overcomes the tension between the goals set out by statisticians, the content concerns of mathematics instructors, and weaknesses of using traditional approaches. The course is part of a National Science Foundation sponsored grant Mathematics and Science Throughout the Undergraduate Curriculum awarded to Indiana University. The NSF grant overcomes cultural impediments to reform, such as reluctance by many teachers to change from traditional lecture type teaching formats. The grant rewards risk taking interdisciplinary approaches, requires reform approaches, and encourages enrolling students who have failed traditional approaches. Mathematics in Action teaches [16] freshmen in Business, Economics, Education, Liberal Arts and Sciences, Nursing, and School of Public and Environmental Affairs. Mathematics in Action uses an interdisciplinary environment in which a diverse group of students and faculty apply learning-centered approaches and engage in discussions. The approach taken in the course resolves the lecturing centered weakness of the traditional approaches. In the course, students learn mathematics by a mixture of traditional instruction and modeling applications. These skills allow students to solve actual service-learning projects for business, industry, social and governmental agencies. By using service-learning projects, Mathematics in Action overcomes the relevance weakness of the traditional freshman level finite mathematics course (MPR goal 5). Organizations, which supply such projects, universally instill in students a sense of the projects' worth. For example, the American Diabetes Association informed students that their efforts would help the organization estimate geographical locations of those in dire need of critical health care services such as testing for diabetes. The relevance issue and lack of statistical content are largely overcome as students involve themselves (MPR goals 1 & 2) in narrowing the problem, struggling to find the relevant data, and attempting to discover, learn, and apply those tools that will lead towards a solution. Conceptual understanding of the traditional core mathematical topics takes place (to satisfy the prerequisite concerns) by

1. Teaching core topics required by replaced traditional courses and
2. Using the tools to solve hands-on service-learning projects and individual projects designed to apply technology [35].

Students use actual data and statistical tools in solving actual business and social problems (Goals 1, 3, 4, & 5 and modeling concerns mentioned by MPR). Thus, the artificiality issue of the applications is resolved. Moreover, students analyze surveys and collect data in a large real engaging environment. Their learning environment extends beyond the classroom to the library, service-learning organization, laboratory, and community. They think statistically and practice statistical reasoning. They use elements such as probabilities, estimations, percents, and computer technology routinely. Students develop skills, in mathematics, statistics, teamwork, data gathering, and communications (oral and written). (MPR goals 3 & 4). Students see the needs for applying technological tools such as Microsoft Excel Spreadsheets to manipulate the data, solve the problem, and present the result (MPR goal 6). This instills an appreciation of the power of technology. They use appropriate technology to solve problems that are unsolvable by more primitive methods.

The use of real data for the projects and the ability to find a solution has highlighted to us the necessity to use computer technology and modern software. We had not envisioned this need the first time we taught the Mathematics in Action course. Initially, we thought that students could use graphing calculators to analyze the data sets for their projects. This was very naive on our part. Many service-learning projects involve large data sets or large numbers of calculations that simply cannot be done on a hand held calculator due to insufficient memory. A delinquency loan rate project for a large financial institution, for example, had approximately 140,000 loan accounts for each of twelve months (longer periods were also available). A routing problem using the traveling salesperson algorithm was too large even for the computer. It had to be broken into parts, and then reconstructed to obtain solutions. This unexpected need for technology turned out to be a blessing in disguise (though this is not necessarily the way we thought about it that first year). The need to use modern day technology has led to an awareness and appreciation by our students of the power of merging technology and mathematics to solving service-learning problems. This has become one of the more important learning outcomes from the use of service-learning projects.

For five years, Mathematics in Action was taught with instruction on technology done through class projects and out of class tutorials. Students complained about the out of class time demands placed on them for a three-credit hour course. This approach also placed a tremendous burden on the faculty teaching Mathematics in Action. Instructors in the course had to not only teach the course, work with five or six organization on projects, coach student teams on the mathematics required for the projects, but on top of all of these, teach students how to use applications software such as Excel, Access, SPSS, Microsoft Word, etc. Prior to coming into this freshman mathematics course virtually none of the students had experience with computer applications. Nonetheless, all students on the IUSB campus are required to take at least one technology course.[2]

Linking Mathematics in Action and computer technology solved two major problems: 1) the linkage reduced student frustration over course time demand. Students would now receive credit for two required courses, making time demands placed on students commensurate with the credits received; and 2) the linkage further lessened the burden placed on

[2]For a detailed discussion of linking introductory mathematics and technology courses see [34]. Projects differ in the tools needed for solutions. For example, with very large data bases Access was first use to link various parts of those data bases so that they could be then used in Excel.

faculty. Each faculty member is given credit for one of the linked courses.[3] This linkage also resulted in several benefits:

1. Mathematical projects were used to illustrate and practice various computer applications;
2. Computer applications skills were used to solve various mathematical problems;
3. The ability to use computer technology to solve large scale problems enhanced the appreciation by students of the power of applied mathematics; and
4. The learning environment due to the linkage provided more opportunities for student collaboration, leading to more effective learning and higher productivity.

These benefits satisfy recommendations made by mathematicians and statisticians concerning preparation of students for entry level statistics and mathematics courses.[4]

In this paper, we describe spreadsheet modeling of various finite mathematics applications designed to solve large scale problems that more closely resemble service- learning problems. Each spreadsheet model is used to reinforce concepts taught in the linked mathematics and technology classes and to help student teams learn how to develop such spreadsheet skills needed for the completion of organizational projects. Furthermore, spreadsheet modeling links and integrates mathematics and technology courses.Through demonstration of mathematical concepts using Excel spreadsheets, we stress synergies between mathematics, technology, and service-learning applications. These synergies emphasize learning goals such as quantitative skill development, analytical and critical thinking, information technology and technological issues, innovative and creative reasoning, and writing across the curriculum.[5]

3. FINITE MATHEMATICS CONCEPTS AND SERVICE-LEARNING PROJECT MODELING USING EXCEL

In a standard finite mathematics course offered for entry level students, students are introduced to topics such as, descriptive statistics and elementary data analysis; counting methods; probabilities including simple, conditional, Bayesian, etc.; systems of linear equations; matrices; systems of linear inequalities; and modeling and optimization problems including linear programming, and the simplex method. In teaching a topic the following sequence of steps takes place:

1. Introduce a problem;
2. Model the problem;
3. Explore multiple approaches to solve the model;
4. Apply algebraic skills while applying a method;
5. Translate the quantitative solutions into meaningful arguments within the context of the original problem;
6. Generalize the model;
7. Apply the method to other problems with other subject matters.

Each topic is practiced by students using small scale problems found in the textbooks and worked out manually. Such small scale problem solving is intended to help students learn and understand the basic principles underling the concepts. However, as discussed earlier, students, in general, consider such textbook type problems artificial and unrelated to service-learning problem solving. Most service-learning situations require defining the problem, data gathering, discovering the appropriate modeling techniques sufficient to solve the problem. Often times such sophisticated modeling techniques require large scale calculations that cannot be done manually. This is the case with the service projects used in the finite mathematics class.[6]. Given that virtually all students have access to computers and take at least one basic computer technology course, it is beneficial to teach linked finite mathematics and computer technology courses. In computer technology part of the linkage, we teach various packages including Microsoft Excel which becomes a modeling and technology environment that allows students to see that with even basic computer spreadsheet skills they can solve large scale service-learning problems. The following demonstrations highlight how Microsoft Excel can be used to enhance learning in both finite mathematics and technology courses, as well as prepare students to tackle service-learning projects involving large, complex data bases; modeling; and cumbersome computations. In each demonstration we show the value of the demonstration to understanding the mathematics principles, the power of Excel as a data analysis and modeling tool, and its application in solving service-learning projects.

4. CALCULATION OF DESCRIPTIVE STATISTICS USING EXCEL

In the Indiana University system, as well as many other schools, finite mathematics is a prerequisite for most majors, especially business, public administration, social sciences, and some life science areas. As mentioned earlier, statisticians MPR (2002) articulate a set of goals for mathematics prerequisites including instilling an appreciation of the power of technology and developing skills necessary to use appropriate technology to solve problems. The service-learning project orientation of Mathematics in Action necessitates that student teams gather, process, analyze, and report on data using various tools acquired in the mathematics and technology courses described above. Descriptive statistics provide a powerful learning tool to introduce students to basic data analysis and presentation, as well as prepare

[3]At Indiana University South Bend, each faculty member receives teaching credit for his/her discipline course when he/she teaches a linked course.

[4]See [3];[23] and [25].

[5]The organizational projects, whose scope requires the use of computer technology skills, link the two courses. Evaluation of the students' learning consists of traditional exams in disciplines, industrial projects, and individual projects. In the mathematics course 44% of the student's grade is for traditional examinations, 40% for team organizational projects, 8% for individual projects, and 8% for portfolio, attendance, etc. In the technology course 70% of the student's grade is for traditional examinations, 22% for projects, and 8% for portfolio, attendance, etc.

[6]See the following articles describing the use of service-learning projects: [2]; [5]; [7]; [8]; [10]; [11]; [14]; [17]; [18]; [26]; [29]; [30]; [34]; [38]; [43]

Table 1: Portion of Sample Data Set for Generating Pivot Tables

Person	Courier	Journal
1	no	no
2	yes	no
3	yes	no
4	yes	yes
5	yes	yes
6	no	yes
7	no	yes
8	yes	no
9	no	no
10	yes	no

Table 2: Excel Generated Pivot Table for Newspaper Example

Count of Persons	Journal		
Courier	no	yes	Grand Total
no	355	258	613
yes	522	365	887
Grand Total	877	623	1500

them for theoretical topics such as probability distributions that could be used for modeling empirical data. Thus, we have found that descriptive statistics is a useful tool for introducing students to many of the upcoming topics in the course.

The mathematics component of the linked courses introduces students to central tendency, dispersion, histograms, and charts using textbook type examples and other examples from popular media (e.g., USA Today, local newspapers, Business Week, etc.) In the linked technology component, students are instructed on how to use Excel spreadsheets to produce the same measures using service-learning actual data bases comparable to those they will work on in their service-learning projects. Such data typically come from personnel data bases that include various characteristics of employees such as gender, schooling, experience, title, salary.[7] We also have generated data bases comparable to those found in various student service-learning projects completed in previous semesters. As an illustration, for example, we created a data set comparable to a similar data base analyzed by a student-team for a local newspaper. The data we generated refer to a situation where 1,500 individuals were randomly called concerning whether they subscribed to the Courier, the Journal, or both newspapers. The first ten observations of that data set are shown in Table 1

This example was used to demonstrate the ability of Excel spreadsheets to take a reasonably large data set and summarize it into a contingency table that can be used to calculate simple and conditional probabilities. Specifically, Excel's pivot table function cross classifies the 1,500 observations as shown in Table 2.

Based on this pivot table, students answer questions such as: What is the probability a randomly selected person subscribes to the Courier (887/1500)? Similarly, what is the conditional probability that a subscriber to the Courier, also subscribes to the Journal (365/887)? Based on this pivot table, students are instructed how to construct tree and Venn diagrams which are useful in describing intersection and unions which underlie the Bayesian formula.

[7]Many of these data bases are found in the textbook used in the computer technology course.

Table 3: Portion of Loan Analysis Project Data

member no	acct no	balance	daysdel	prod code
4001026	155	8090.38	60	3
5001026	156	9999.99	37	3
3201027	145	5343.35	46	41
6601087	166	10046.19	31	60
7772032	145	7662.47	15	35
1242420	155	202.18	10	5
9722423	155	4627.82	15	6
5822551	141	672.76	31	50
3072551	155	6153.57	7	5

Excel 2007 provides a Venn diagram which helps a user convert the Pivot table into a graphic form. The following example converts the data in Table 2 into the Venn diagram shown in Figure 1.

This Venn diagram helps students visualize concepts such as union, intersection, and conditional probabilities.

As part of the instruction in descriptive statistics, students also make frequency distributions. Most textbook examples employ relatively small data bases (e.g. usually fewer than 50 observations) so that the frequency distributions can be made manually using tallying methods. Service-learning projects usually involve data sets consisting of large numbers of observations making manual approaches time consuming and subject to error. For example, in one of our service-learning projects for a local bank concerning delinquent loans, each month's delinquent loan data base contained approximately 7,000 loans, all in different stages of delinquency. A small segment of that data base is given in Table 3.

Table 3 contains 10 of the 6904 delinquent accounts. The first two columns represent the member number (which we have modified for security reasons) and the loan number. The next two columns contain the loan amount and days delinquent, and the final column contains a product code (e.g. auto, personal, home, etc.). Students were instructed on how to create a frequency distribution using Excel. Part of the instruction involved selecting meaningful intervals. After discussions with the client-bank, intervals of 1 to 30, 31-60, etc. were selected. Excel's Frequency function generated Table 4.

Excel requires as inputs the upper level of each interval (bins) with an empty interval capturing all observations greater than the largest interval. The format of the Excel output is not very meaningful for a general audience. Students thus are instructed to modify the Excel generated table to clarify the information. Table 5 represents a more user friendly version of Table 4.

Furthermore, the same example is used to produce the histogram shown in Figure 2.

As illustrated in Table 6, the above examples can easily be extended to introduce basic empirical probability concepts which can be easily computed from either the frequency dis-

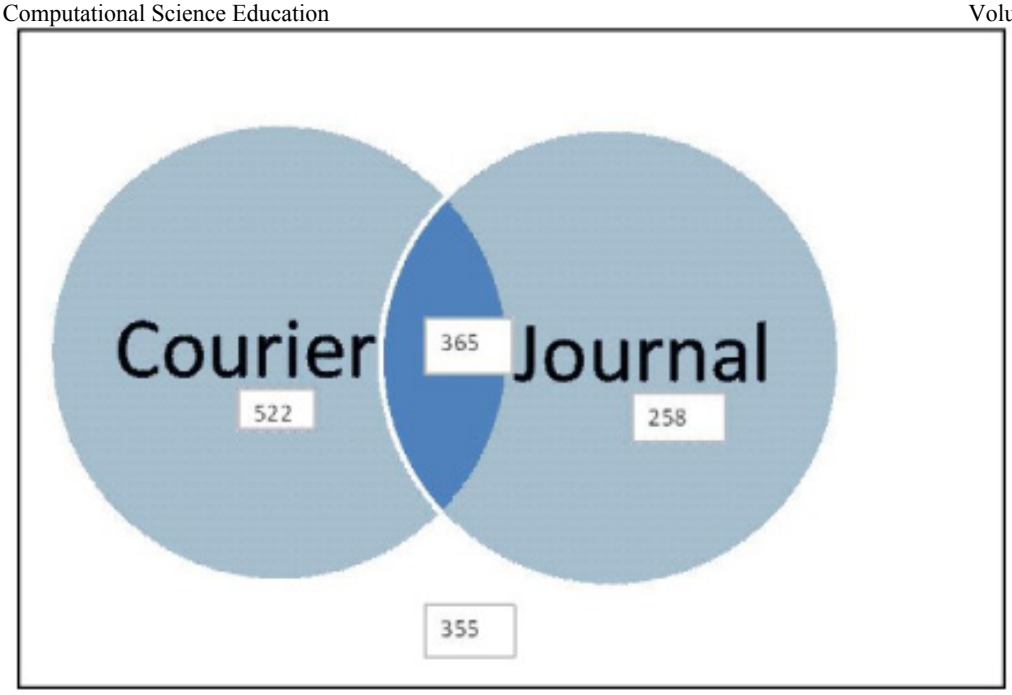


Figure 1: Excel Generated Venn Diagram Based On Pivot Table

Table 4: Excel Generated Frequency Distribution from Loan Analysis Data

bins	Number
30	3739
60	2082
90	466
120	220
150	112
180	96
	189
Total	6904

Table 5: Reformatted Excel Frequency Distribution

Days Deliquent	Frequency
0-30	3739
31-60	2082
61-90	466
91-120	220
121-150	112
151-180	96
over 180	189
Total	6904

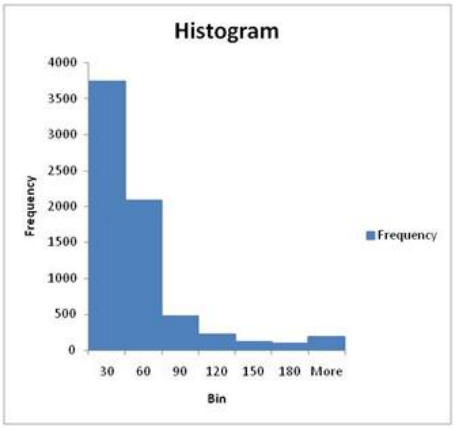


Figure 2: Histogram Based On Bank Loan Frequency Distribution

Table 6: Empirical Probabilities Calculated from Excel Frequency Distribution

Days Deliquent	Frequency	Empirical Probabilities
0-30	3739	0.542
31-60	2082	0.302
61-90	466	0.067
91-120	220	0.032
121-150	112	0.016
151-180	96	0.014
over 180	189	0.027
Total	6904	1.000

tribution or the histogram.

This example can be further used to illustrate generating research questions and answering such questions with contingency tables and tree-diagrams. An example based on the loan data is given in Table 7 and Figure 3.

Contingency Table 7 was produced using the Pivot function discussed earlier to answer a research question about whether the bank anticipates slow paying borrowers by charging a higher interest rate on the loan. The answer is clearly yes. For instance, the simple probability of a loan being bad (loans being delinquent more than 30 days) is .0103. However, for customers receiving the very lowest rates, and presumably judged to be the best credit risk, the probability of a bad loan is virtually zero. In contrast, for customers paying the very highest rates, and again presumably judged to be poor credit risks, the probably of a bad loan increases to 0.0296, almost triple the simple probability of a bad loan. Continuing with this example, one also can produce the tree diagram given in Figure 3.

Given information about the rate a borrower pays, Bayes Theorem can be used in conjunction with either the above Table 7 or the tree-diagram given in Figure 3 to find the probability that the loan is either good or bad. For instance, the student-team working on this service- learning project observed that the contingency table and the calculations from Bayes formula using information in the tree-diagram agreed, namely that when the bank charged a rate greater than 10 percent, the probability that the loan ultimately would be bad was 0.0296. We found it to be very successful as a teaching approach to employ multiple approaches to illustrating complex topics, such as Bayes Theorem.

In addition to the Pivot Table function described above, Excel's filtering and sorting capabilities make it possible using a large data base to investigate research questions related to how customer and/or loan characteristics contribute to defaults. In the local bank service-learning project described above, the client was interested in the characteristics of customers, loans, or both that contributed to higher delinquency probabilities. This led the student team to sort and filter the data. The student team used Excel to sort selected data by loan size to investigate the characteristics of delinquent loans based on whether they were large, medium, or small. Also, the student-team filtered the data by product type (e.g., car loans, personal loans, home mortgages, and the like) to see whether loan delinquency probabilities depended on loan type. Further, the student-team filtered the data by the income of loan recipients, as well as their credit report characteristics (e.g., credit score, number credit cards, past delinquent accounts, and the like). This filtering and sorting provided a powerful tool for preliminary causal analysis of the determinants of loan delinquency rates that might be used as a decision tool for designing future lending products and offering loans to future potential customers.

5. PERMUTATIONS AND COMBINATIONS

As an introduction to probabilities, students are introduced to basic counting methods. Combinations and permutations represent a universal counting approach. The computations for these techniques can be quite cumbersome. Although some more advanced hand held calculators provide options for calculating combinations and permutations, they are, nonetheless, much easier using Excel. In addition, most students once they start working will have more access to Excel than to any other means of computing. Excel provides formulas that can be inserted into a document so that repetitive computations can be quickly made. Part of the teaching in finite mathematics involves practicing the use of the formulas for combinations and permutations. We use various examples related to selecting, for example, 3 persons from a group of 10 to serve on a committee. One type of selection is where the order of selection is not important; the alternative selection makes the first person president, the next vice president, and third treasurer. Students learn the combination and permutation formulas to calculate the number of possible selections in each case. Furthermore, these numbers could be used for calculating probabilities, such as the probability that a three-person selection would have at least one female or that a particular arrangement happens. There are many other applications related to quality control, gambling, statistical sampling, and the like.

The Excel example given in Figure 4 attempts to interconnect what students learn in the mathematics class and what Excel's spreadsheet provides in terms of a computational environment.

As shown in Figure 4, there are 254, 251, 200 permutation and 2,118,760 combinations of five items taken out of a total 50 are calculated. By simply changing the number of objects and number of objects to be selected in the upper left hand box, students can quickly find permutations and combinations. These formulas have direct application in determining probabilities such as quality control problems. A common problem (using the techniques shown in Figure 4) given in our textbook asks the probability of having defective calculators in a sample of five chosen from a box of 30 calculators purchased by a school of which historically 10 percent are defective. The easiest way to solve this is to use the complementary problem which calculates the probability of no defectives in a sample of five and then subtracts this result from one. In terms of the formulas defined in Figure 4, the probability is obtained as:

$$\begin{aligned} P_{(\text{at least 1 defective from a sample of 5})} &= 1 - P_{(\text{0 defectives})} \\ &= 1 - \frac{C_{27,5}}{C_{30,5}} \\ &= 1 - \frac{80,730}{142,506} = 0.433 \end{aligned}$$

where $C_{27,5}$ is the number of samples of size 5 with no

Table 7: Research Questions Formulated From Excel Loan Analysis Pivot Table

Research Question: Is there a relationship between the annual percentage rate and days delinquent?					
		Annual Percentage Rate			
		<5	>=5<10	>=10	**Total**
Days Delinquent	Good	24	12706	4859	17589
	Bad	0	35	148	183
	Total	24	12741	5007	17772

P(Good)	0.989703	P(Bad)	0.0103
P(Good|<5)	1.000	P(Bad|<5)	0.000
P(Good|>=5<10)	0.9973	P(Bad|>=5<10)	0.0027
P(Good|>=10)	0.9704	P(Bad|>=10)	0.0296

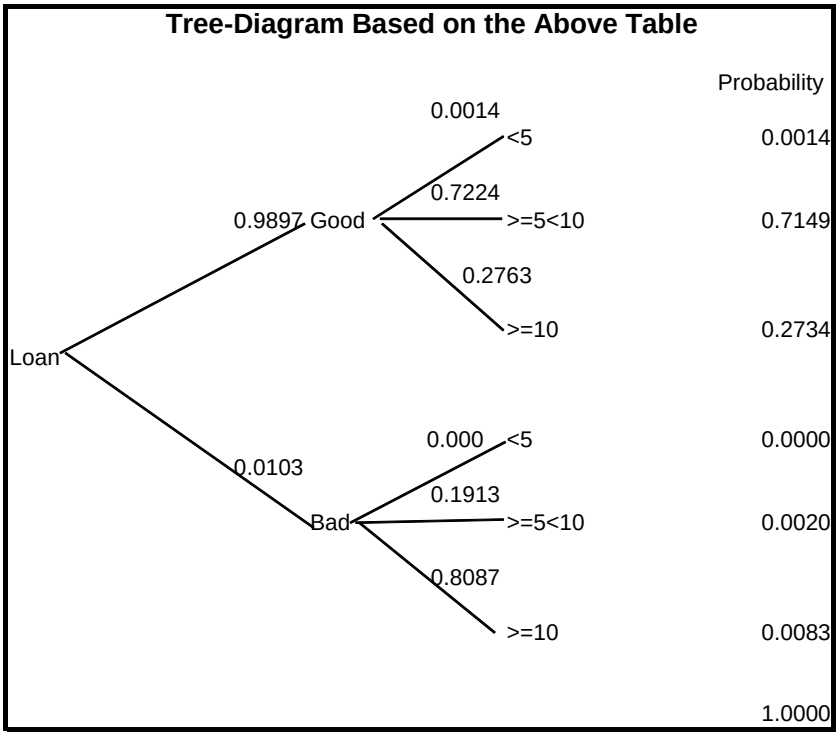


Figure 3: Tree-Diagram Based On Excel Pivot Table

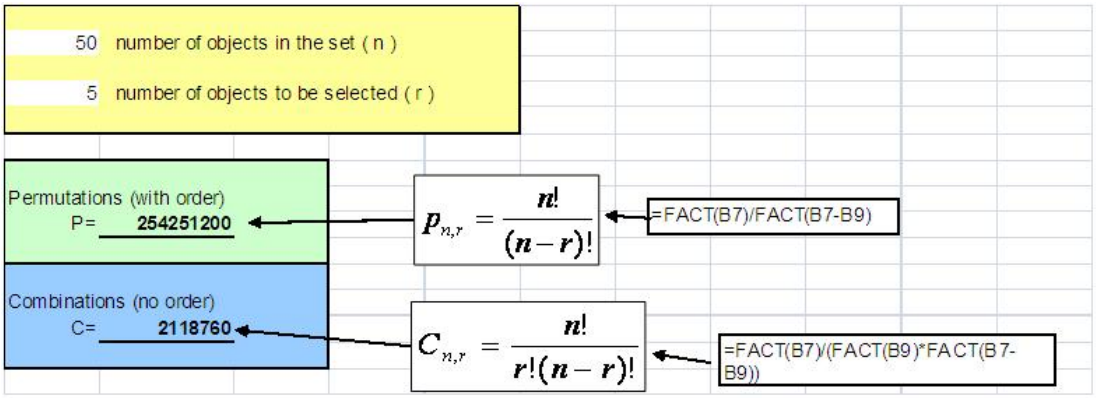


Figure 4: Excel Calculations of Permutations And Combinations

defective calculators and $C_{30,5}$ is the number of samples of size 5 from the population of 30. There are many other similar quality control problems as well as classical examples using decks of cards (e.g., the probability of obtaining a "full-house" in a five card poker hand).

The use of combinatorial techniques played a role in helping students understand one of our earliest service-learning projects that dealt with finding the most efficient lunch delivery routing system for a local school system, a classic Traveling Salesperson problem. Resources became routes between school buildings, costs distances of these routes, with the objective to minimize the total cost of the route. Constraints required that each location be visited once and only once. The analysis required the student-team to gather information on distances (or travel times) between various school buildings for the school corporation's internal delivery system.

We created a simplified version of the problem, for instance for four cities, so that they could see just what was involved. For example, suppose a salesperson starts from a corporation in Chicago and wants to visit the cities South Bend, Indianapolis, and Michigan City. Distances between these cities are found in Table 8.

Combinatorial method calculations indicate that there are six possible paths listed in Table 9, for each path the distance was calculated by hand from data in Table 8.

The optimal solution are the following routes with the minimum distance 375 miles (Chicago, to Michigan City, to Indianapolis, to South Bend, returns to Chicago. Alternatively, follow this path in the reverse order).

Chicago, to Michigan City, to Indianapolis, to South Bend, returns to Chicago. Alternatively, follow this path in the reverse order.

We used Excel to demonstrate to the students that as the number of destination points increased the permutations that would require evaluation became extremely large. For illustration, if there are 12 points instead of 4 points, the combinatorial solution will involve 11! or 39,916,800 possible patterns which are too many to evaluate manually.

Our Excel demonstrations using combinatorial techniques led students to investigate other more efficient solution algorithms such as binary programming and Excel's solver module.

6. SIMULATION OF STOCHASTIC PROCESSES

In teaching probability concepts, students often confuse theoretical and empirical probabilities. For example, the probability of a head when tossing a fair coin is 0.50 while tossing a fair coin 100 times may lead to far more or far fewer than 50 heads, perhaps even as many as 70 or as few as 30. Most students perceive that a theoretical probability of 0.5 means that nearly (if not exactly) 50 of the 100 flips should be heads. Such confusion can be dealt with by setting up experiments that compare empirical and theoretical probabilities. Prior to using Excel, we had each student toss a coin 10 times and tallied results for all students. The procedure is both time consuming and limited to a small number of tosses. For efficiency and flexibility, we use the power of Excel to simulate stochastic processes. The example used for illustration is that of rolling a six-sided die 1000 times. Once the spreadsheet is setup we can simulate 1000 rolls (or any other number) over and over and then compare the empirical outcomes to the theoretical probabilities. The same procedure could be used for tossing coins, rolling more than one die, dealing card hands, spinning of a roulette wheel, and the like.[8] In the illustration shown below, in addition to having Excel simulate the roll of a die 1000 times we also construct a frequency distribution and histogram of outcomes to facilitate comparison of empirical and theoretical probabilities. The simulation output displayed in Table 10 is produced by using the formulas `ROUNDDOWN(6*RAND(),0)+1` and `FREQUENCY(C4:C1003,F4:F9)`, where `C4:C1003` contain the 1000 randomly generated die numbers and `F4:F9` hold the die values 1 through 6.

By hitting the F9 key in Excel, students see how each outcome of 1000 rolls differs from each other and from the theoretical probabilities. This simulation would provide tools for more advanced discussions of stochastic processes dealing with the law of large numbers, the central limit theorem, confidence intervals, and the like.

A number of our service-learning projects involved stochastic processes. In one such service-learning project, students were asked to make recommendations on whether it would

[8]As the number of trials increases, the empirical probability distributions converge to the theoretical distributions.

Table 8: Hypothetical Traveling Salesperson Routing

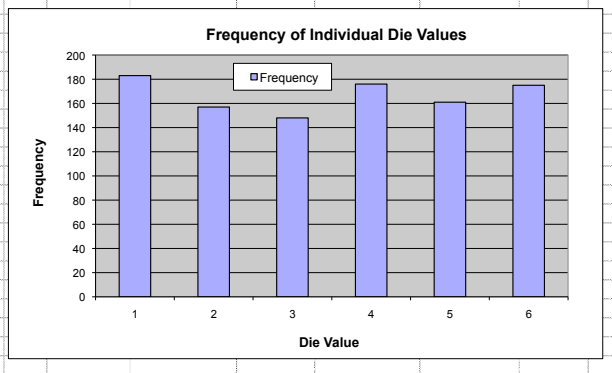
	1-Chicago	2-South Bend	3-Michigan City	4-Indianapolis
1-Chicago	0	90	60	150
2-South Bend	90	0	40	135
3-Michigan City	60	40	0	160
4-Indianapolis	140	130	160	0

Table 9: Possible Routing Patterns Based On Hypothetical Example

Route	Total Distance
1-2-3-4-1	440
1-2-4-3-1	445
1-3-2-4-1	375
1-3-4-2-1	445
1-4-2-3-1	375
1-4-3-2-1	440

Table 10: Excel Simulation of Rolling A Die 1000 Times

Die Value			Die Value	Frequency	Rel. Freq	Theoretical	Difference		
6			1	191	0.191	166.67	24.33	Mean	3.425
6			2	160	0.16	166.67	-6.67	Median	3.00
1			3	166	0.166	166.67	-0.67		
4			4	165	0.165	166.67	-1.67		
1			5	152	0.152	166.67	-14.67		
6			6	166	0.166	166.67	-0.67		
2				1000	1	1000.00	0.00		
2									
4									
2									
3									
3									
6									
5									
6									
1									
1									
1									
3									
1									
2									
3									
2									
6									
2									
3									
6									
2									
3									
6									



be less costly to change light bulbs in an elementary school as they burned out or to periodically change all the lights in the school at the same time. The hours of life of a light bulb follows a stochastic process generally modeled with the exponential distribution. For example, a florescent light bulb, with an expected life of 30,000 hours, has a cumulative probability distribution given in Figure 5 where the horizontal axis represents the life of a bulb.

The student-team working on this project recognized the tradeoff between higher maintenance costs if bulbs are replaced as they burn out versus higher light bulb costs if bulbs are all replaced at the same time. The stochastic bulb life process suggested that about 10 percent of the bulbs would have a life of approximately 4,000 hours. If 10 percent burnt out bulbs was an upper limit on the acceptable number of bulbs burnt out at any one time, then about 90 percent of the bulbs would be discarded far before they burned out. Indeed, about one-half of the bulbs would last more than 20,000 hours. The students thus estimated the costs of the two strategies and concluded that "Taking into consideration the amount of money that would be lost with the implementation of any of these total bulb-changing policies, we recommend that the [the client] continue their current policy of changing light bulbs on an as-needed basis."

7. MODELING OPTIMIZATION PROBLEMS WITH CAPACITY CONSTRAINTS

A key concept students learn in the linked finite mathematics and technology courses is the value of solving constrained optimization problems. Not only are these concepts and solution techniques valuable to business and economics students, but they also play important roles in several of the service-learning projects undertaken by student teams. As an illustration, a student-team applied constrained optimization techniques to find an optimal mix of fund raising activities for a local chapter of the American Diabetes Association (ADA). Basically, the local ADA director had limited resources such as working capital, hours of volunteer time, restrictions on her own time, frequency of certain types of fund raising events (e.g., only so many fund raising walks can take place in a given year), and the like. The student team undertaking this project gathered data on estimated revenues from each event, as well as the amount of working capital, volunteer time, etc. each event required. After solving the problem for the optimal portfolio of events, students then performed post- optimality sensitivity analysis influenced by weather conditions. Fundraising walks, for example, raise much smaller amounts on cold unpleasant days than under more favorable conditions. The student team indentified three possible post-optimality scenarios: 1) normal (average) weather conditions; 2) exceptional (above average) weather conditions; and 3) severe (below average) weather conditions. Other student teams also applied these techniques to optimal land allocation for a mall parking lot, optimal use of full-time v. part-time workers for a local school corporation, optimal routing for a school corporation lunch delivery program (the traveling salesperson problem mentioned above), and the optimal keep v. replacement policy for school corporation trucking equipment.

As part of the curriculum, we teach students the mathematics behind optimization with various capacity constraints as well as how to model such problems using Excel spreadsheetss and the Excel solver module. For example, we use problems from our course textbook[9]. One such problem is given below: Construction-resource allocation. A contractor is planning to build a new housing development consisting of colonial, split-level, and ranch-style houses. A colonial house requires 1/2 acre of land, $60,000 capital, and 4,000 labor-hours to construct, and returns a profit of $20,000. A split-level house requires 1/2 acre of land, $60,000 capital, and 3,000 labor-hour to construct, and returns a profit of $18,000. A ranch house requires 1 acre of land, $80,000 capital, and 4,000 labor-hours to construct, and returns a profit of $24,000. The contractor has available 30 acres of land, $3,200,000 capital, and 180,000 labor-hours.

A How many houses of each type should be constructed to maximize the contractor's profit? What is the maximum profit?

B A decrease in demand for colonial houses causes the profit on a colonial house to drop from $20,000 to $17,000. Discuss the effect of this change on the number of houses built and on the maximum profit.

C An increase in demand for colonial houses causes the profit on a colonial house to rise from $20,000 to $25,000. Discuss the effect of this change on the number of houses built and maximum profit.

Prior to solving the problem, students are shown how to model the problem in an Excel spreadsheets. Table 11 displays the organizational spreadsheet for part B of the problem.

The spreadsheet includes formulas that sum up the products of the decision variables and per unit resource usage amounts as well as total profits. For example, the total usage of land is computed using the Excel formula `SUMPRODUCT(F6:H6,$E$16:$G$16)`; where `F6:H6` are cells containing per unit land resources for colonial, split- level, and ranch homes, respectively, and where `E16:G16` represent the numbers of each type of produced house. This model allows us to discuss trial and error approaches to solving the feasible and optimal mix of houses. For example, the spreadsheet in Table 12 shows the profits and resource utilization for producing 20 of each type of house. Although total profits are high, the solution is not feasible since all of the capacity constraints are violated. The spreadsheet model makes it very easy to test numerous combinations of the three types of houses and to discuss whether or not they are feasible. Even when feasible outcomes are obtained there is no insurance that these outcomes are optimal. Students quickly learn that a trial and error approach which could conceivably list every possible outcome is not only time consuming but very inefficient. Besides exploring the solution to this problem using the Simplex Method and graphical techniques, we teach students in the technology course how to use the Excel solver to find the optimal solution.

Numerous other examples are used dealing with manufacturers, nutrition, portfolio allocation, and the like. An example showing the entire Excel spreadsheets formulation for minimizing the transportation cost taking students and chaperones on a trip by two different means of transportation is found in Figure 6.

[9]See [1], problem 43, p. 314.

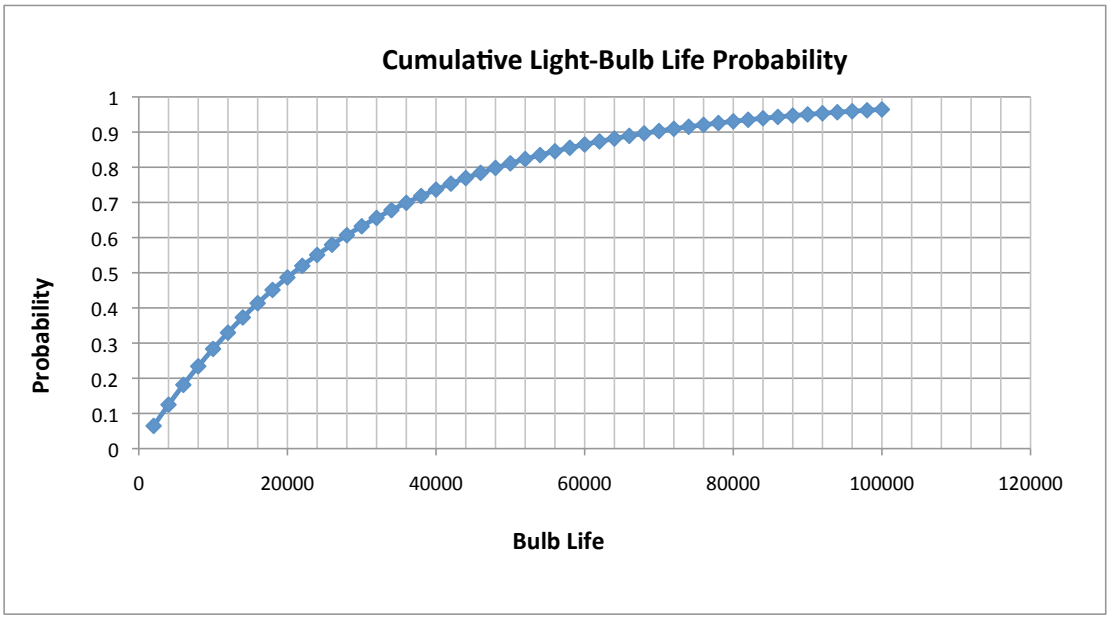


Figure 5: Excel Generated Exponential Probability Distribution

Table 11: Excel Spreadsheets for Modeling Linear Programming Problem

	C	S	R	USEAGE		CAPACITY
LAND PER HOUSE (ACRES)	0.5	0.5	1	0	<=	30
LABOR PER HOUSE (HOURS)	4000	3000	4000	0	<=	180000
CAPITAL PER HOUSE ($)	60000	60000	80000	0	<=	$3,200,000
				TOTAL PROFIT		
PROFITS PER HOUSE	$17,000	$18,000	$24,000	$0		
ACTIVITIES	NO.C	NO.S	NO.R			
DECISION VARIABLES	0	0	0			

C=COLONIAL, S=SPLIT-LEVEL,R=RANCH

Table 12: Excel Trial and Error Demonstration of the Construction Problem

	C	S	R	USEAGE		CAPACITY
LAND PER HOUSE (ACRES)	0.5	0.5	1	40	<=	30
LABOR PER HOUSE (HOURS)	4000	3000	4000	220000	<=	180000
CAPITAL PER HOUSE ($)	60000	60000	80000	4000000	<=	$3,200,000
				TOTAL PROFIT		
PROFITS PER HOUSE	$17,000	$18,000	$24,000	$1,180,000		
ACTIVITIES	NO.C	NO.S	NO.R			
DECISION VARIABLES	20	20	20			

C=COLONIAL, S=SPLIT-LEVEL,R=RANCH

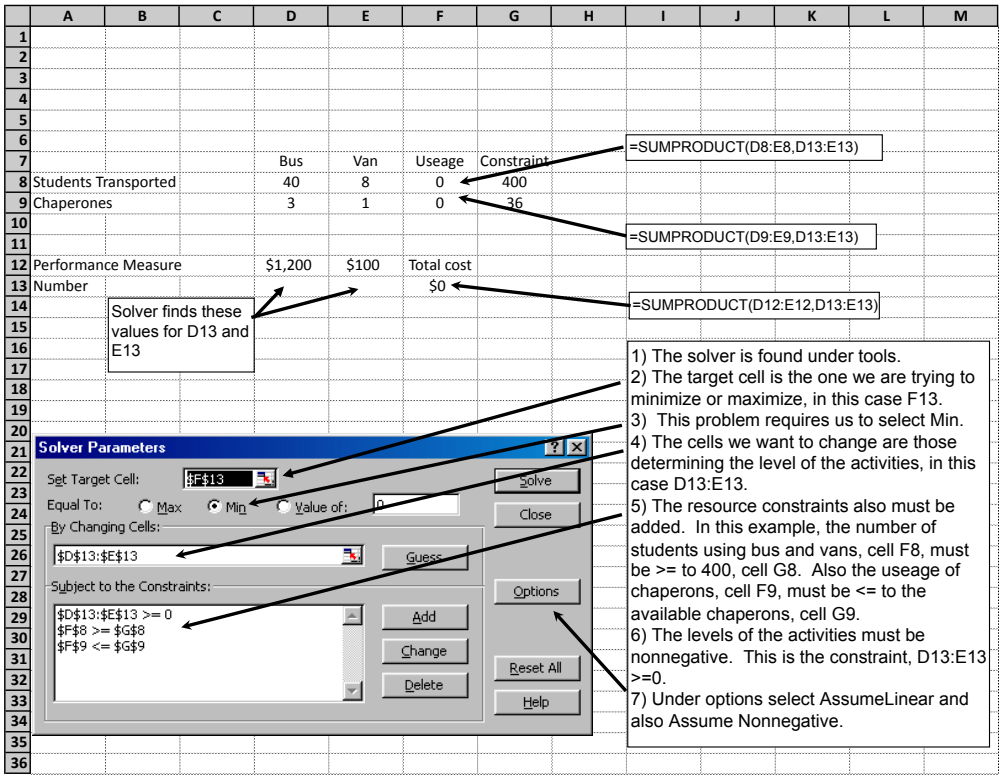


Figure 6: Excel spreadsheets for Solving Transportation Example

Table 13: Excel Solver Module Solution to Ada Event Mix Problem

ADA EVENT MIX LINEAR PROGRAMING AFTER USING SOLVER																
			Walk SB	SP Wk CH	SP Wk RC	SP Wk GS	SCH WK	KS-PIG-SB	KS-PIG-LP	WINE	TREES	K	L	USEAGE		CAPACITY
DIRECTOR'S TIME OPERATING			32	53	58	48	20	25	25	18	20	-1		0.000	=	0
DIRECTOR'S TIME SET UP			560	142	142	132	54	115	90	88	40	1		2357.000	<=	3000
VOLUNTEER TIME			50	120	120	120	48	72	72	40			-1	0.000	=	0
CAPITAL			21000	8750	5433.75	7000	1250	3450	3990	7650	31.25			73626.250	<=	75000
WALK SB CONSTRAINT			1											2	<=	2
WALK CH CONSTRAINT				1										1	<=	2
WALK RC CONSTRAINT					1									1	<=	2
WALK GS CONSTRAINT						1								0	<=	2
WALK WK CONSTRAINT							1							2	<=	2
SCH WALK CONSTRAINT								1						2	<=	2
KISS-PG SB CONSTRAINT									1					2	<=	2
KISS-PG SB CONSTRAINT										1				0	<=	1
WINE CONSTRAINT											1			2	<=	2
TREES CONSTRAINT																
REVENUE			60000	25000	15525	20000	5000	22000	26600	17000	625					
COST			12175	5950	4290	5950	1140	2020	1615	6400	100					
PERFORMANCE			47825	19050	11235	14050	3860	19980	24985	10600	525	0	0			
ACTIVITIES			Walk SB	SP Wk CH	SP Wk RC	SP Wk GS	SCH WK	KS-PIG-SB	KS-PIG-LP	WINE	TREES	K	L	NET SURPLUS		
NUMBER			2	1	1	0	2	2	2	0	2	355	724	224635		

The American Diabetes Association example mentioned above provides an excellent example of how students can generalize the Excel examples to a service-learning project. After collecting the appropriate data and using Excel's solver, the student-team working on this project generated the output found in Table 13.

Given the various constraints on the director's time, volunteers' time, working capital, and the maximum number of times an event could be held in a given period, the student-team was able to recommend the mix of fundraising events that would maximize the net fundraising surplus (e.g., two SB walks, one spring walk CH, one spring walk RC, etc.). In addition, the student-team was able to test the sensitivity of the outcomes to various assumptions about weather conditions and therefore introduce a stochastic element into their recommendations.

8. CONCLUSIONS

The Excel demonstrations given above promote synergies between mathematics, technology, and applied research. We have found that students enjoy the hands-on nature of the Excel software and engage with it enthusiastically in working on their service-learning projects. Indeed, student comments such as those given below attest to the strong appeal this approach offers.[10]

> "We actually got to use things like probability and frequency tables and see that it worked. We were not just taking a test."

> "I don't have a fear of math now. I understand where you can utilize it in everyday life."

> "I was never really good in math, this course gave me courage; I'm not afraid of it anymore. I felt safe and secure in coming to class... This type of class would have a whole generation of people loving math."

> "We got to work, every week or so, interactively with the business leaders, working on an active problem they are having with their business."

> "This was a great hand-on experience that took my fear out of working within a business and handling specific problems and finding a solution where possible. It showed us that no matter what the problem is, there is a formula that can help to work it out for the best of the business and employees."

> "Having a real -life problem to handle and solve left a more realistic impression of what the work world has to contend with to do the best job you can for your business, employees, and customers."

> "I learned a lot about math as well as the business world."

> "I like unique way of learning math and relating it to real world."

> "I enjoyed working in a group to help solve a common problem. The hands-on experience of working in a real-life situated problem was a great opportunity to have. I truly appreciate the instructors, business and the developers of the program for allowing helping and encouraging us to perform at a higher level."

> "I liked the hands-on, interactive projects we had to do. Having the availability of two professors and one tutor helped immensely. Doing the projects will solidify what I have learned."

The Excel demonstrations also appear to help students better grasp the mathematical principles underlying the demonstrations and to better appreciate the power of applied mathematics and statistics in investigating research questions using actual real-world data bases. The Excel applications directly achieve two of the goals mentioned earlier that statisticians propose for introductory mathematics classes: 1) use real, engaging applications through which students can learn how to draw connections between the language of mathematics and the context of the application; and 2) instill appreciation of the power of technology and develop skills necessary to use appropriate technology to solve problems, develop understanding, and explore concepts. It is our observation that students taking a course that links, mathematics, technology, and service-learning projects not only prepare themselves to undertake service-learning projects in their introductory mathematics course but further take away from their experiences mathematical, technological, and research skills that they then apply in their future courses in business and economics. Equally important, students leave the courses with a new appreciation of the power of mathematics and technology. This is the very outcome the National Science Foundation hoped for when funding service-learning courses such as Mathematics in Action.

[10]The above comments are from students that were made in the students' course evaluation, an interview made by a grant administrator, and newsletters. Instructors were not present when these comments were made. Because the course was an experimental course attempting to demonstrate to beginning students the power of mathematics to solving service-learning projects, the National Science Foundation undertook in-depth qualitative evaluations of students. In addition, a School of Education colleague of ours also interviewed students concerning various aspects of their experience. The comments provided above are typical of these evaluations. The value of using Excel demonstrations enhanced the service-learning project reports and made possible sophisticated data analysis. For example, because of their acquired skills in using Excel, students were able to successfully complete service-learning projects such as the traveling-sales person problem, contingency table analysis of delinquent loans and borrower characteristics, linear programming solutions to fund-raising activities, probability distribution analysis leading to efficient light bulb replacement policies, and a host of other complex problems. Evaluators of the course thought these were extraordinary accomplishments for entry level students.

9. REFERENCES

[1] Barnett, R. A. and M. E. Ziegler. 2008. *Finite Mathematics for Business, Life Sciences, and Social Sciences, 11th edition*, Pearson/Prentice Hall, Upper Saddle River, New Jersey.

[2] Bixby, J.A., J.R. Carpenter, P.L. Jerman, and B.C. Coull. 2003. Ecology on campus. *Journal of College Science Teaching 32* (5): 327-331.

[3] Cobb, G.W. 1992 , "Report of the Joint ASA/MAA Committee on Undergraduate Statistics," *Proceedings of the Section on Statistical Education American Statistical Association*, 281-283.

[4] COMAP 2009. *For All Practical Purposes, 8th Edition*, W. H. Freeman and Co., New York.

[5] Engel, A., C.L. May, and M. O'Leary. 2005. "The Baltimore City Fire Department staffing problem." *In Hadlock 2005*, 35-46.

[6] Grinshpan, J.R. 2005. "The mathematics umbrella: Modeling and education." *In Hadlock 2005*, 47-58.

[7] Grossman, Julie, and Terrence Cooper. 2004. "Linking environmental science students to external community partners: A critical assessment of a service learning course." *Journal of College Science Teaching 33* (5) (March-April): 1-5.

[8] Hadlock, Charles R. ed. 2005. Mathematics in Service to the Community: Concepts and Models for Service-Learning in the Mathematical Sciences. *Mathematical Association of America*, Washington, DC

[9] Hatcher T., Hinton B., et. al. 1996. Graduate Student's Perceptions of University Team Teaching. *College Student Journal*, *Vol. 30*, Issue 3, p. 367.

[10] Hydorn, Debra, L. 2005. Community service projects in a first statistics course. *In Hadlock 2005*, 111-122.

[11] Ing, P.H. 2005. Designing efficient snow plow routes: A service-learning project. *In Hadlock 2005*, 69-80.

[12] Jerde, C.L. and M.L. Taper. 2004. Preparing undergraduates for professional writing. *Journal of College Science Teaching 33* (7): 34-37.

[13] Kochanowski, P. and Shafii-Mousavi, M. 2003a. Mathematics Preparation for First Statistics Courses. *2003 Proceedings, American Statistical Association*. 2205-2212.

[14] Kochanowski, P. and Shafii-Mousavi, M. 2003b. Project Based Learning in an Interdisciplinary Economics and Mathematics Service Course. *Midwest Business Economics Association Proceedings*. 161-169.

[15] Kochanowski, P. and Shafii-Mousavi, M. 2000. How to Design and Teach a Project Based First-Year Finite Mathematics Course. *UMAP Journal*, *COMAP*, Vol. 21.1 Summer, p.119-138.

[16] Kochanowski, P. and Shafii-Mousavi M. 1998. Lessons Learned From Team Teaching an Interdisciplinary Introductory Course in Economics and Mathematics, *Proceedings of the Mathematical Modeling in the Undergraduate Curriculum*, University of Wisconsin - La Crosse, 1998.

[17] Maki, D., Winston, W., Shafii-Mousavi, M., Kochanowski, P., Lang, C., Ernstberger, K., and Hodgson 2005. On the Use of Client Projects in the Mathematics Classroom. *Primus: Problems, Resources, and Issues in Mathematics Undergraduate Studies*. United States Military Academy, West Point, New York 10996-9902 USA. Forthcoming.

[18] Massey, M. 2005. Service-learning projects in data interpretation. *In Hadlock 2005*, 123-130.

[19] McIntosh, M. and Johnson, D. L. 1994. An Instrument to Facilitate Communications between Prospective Team Teachers. *Clearing House*, *Vol. 67* Issue 3, p. 152.

[20] McKelvey, S. 2005. Real-world consulting: The Saint Olaf mathematics program. *In Hadlock 2005*, 25-34.

[21] Michaelsen, L. K. 1999. Integrating the Core Business Curriculum an Experienced Based Approach. *Selections*, *Winter*, pp. 9-17.

[22] C. D. Miller, V. E. Heeren, and John Hornsby. 2004. *Mathematical Ideas, 10th Edition*, Pearson/ Addison Wesley, Boston, Massachusetts

[23] Moore, T., Peck, R., and Rossman, A. 2002. "Statistician Advise Mathematics Association on Undergraduate Curriculum," *Amstat News.*

[24] G. L. Musser, Burger W.F and Peterson B. 2008. *Mathematics for Elementary Teachers, 8th Edition*, John Wiley & Sons Inc.

[25] O'Shea, D. and Pollatesek, H. 1997. "Do We Need Prerequisites?," *Notices of the AMS*, 564-570.

[26] Phillips, M.W. 1997. Teaching general biology for nonmajors through community projects. Journal of College Science Teaching *26* (4): 253-257.

[27] Poff, J.M., D. Larson, and C.F. Rodell. 2004. Biology of the Southwest. *Journal of College Science Teaching 33* (6): 40-44.

[28] Ramsay, J.R. 2005. Creating experience in an experiential learning. *In Hadlock 2005*, 47-58.

[29] Reed, G. 2005. Perspectives on statistics projects in a service-learning framework. *In Hadlock 2005*, 83-88.

[30] Roberts, Catherine A. 2005. Perspectives on modeling applications in a service learning framework. newblock *In Hadlock 2005*, 13-24.

[31] Root, R., T. Thorme, and C. Gray. 2005. Making meaning, applying statistics. *In Hadlock 2005*, 89-100.

[32] Schrage, L. 1986 LINDO (Linear Interactive and Discrete Optimizer) LINGO. *It is a user-friendly computer package that can be used to solve linear, integer, and quadratic programming problems*.

[33] Shafii-Mousavi, M. and Kochanowski, P. 2006. "Integrating First-Year Technology And Finite Mathematics Courses". *PRIMUS Problems, Resources, and Issues in Mathematics Undergraduate Studies*, *Vol. XVI(1)*, pp. 61-80.

[34] Shafii-Mousavi , M. and Kochanowski, P. 2006 "Service-Learning Projects in Linked Mathematics and Computer Technology Courses," *Umap*, pp. 449-468.

[35] Shafii-Mousavi , M. and Kochanowski, P. 1999. The Use of Computer Technology in a First-year Finite Mathematics Course. *Proceedings International Conference on Mathematics/Science Education & Technology*. Charlottesville, Virginia Association for the Advisement of Computing in Education.

[36] Shafii-Mousavi, M. and Kochanowski, P. 1998a. MATHEMATICS IN ACTION: Social and Industrial Problems. *1998 Mathematical Modeling Symposia. Proceedings*. University of Wisconsin - La Crosse,

Wisconsin

[37] Shafii-Mousavi, M. and Kochanowski, P. 1998b. http://oit.iusb.edu/~mshafii/math-in-action.html

[38] Sungur, E.A., J.E. Anderson, and B.S. Winchester. 2005. Integration of service learning into statistics education. *In Hadlock 2005* 101-110.

[39] Watkins, T.L. 1996. Creating a New MBA Core with Team Teaching. *Journal of Management Education Vol. 20*, Issue 4, p. 411.

[40] Webster, J., and C. Vinsonhaler. 2005. Getting down to work– A "how-to" guide for designing and teaching a service- learning course. *In Hadlock 2005* 247- 264.

[41] Wilke, R.R. 2002. Practical considerations for assessing inquiry-based instruction. *Journal of College Science Teaching 31* (7): 432-435.

[42] Winston, W.L. 1994. *Operations Research* Wadsworth Publishing Company, Belmont, California

[43] Wood, B. 2003. Improving wellness on campus. *Journal of College Science Teaching 33* (2): 2731.

[44] Young, M. B. and Kram, K. E. 1996. Repairing the Disconnects in Faculty Teaching Teams. *Journal of Management Education 20*, Issue 4, p. 500.

Computational Chemistry for Chemistry Educators

Shawn C. Sendlinger
North Carolina Central University
Department of Chemistry
1801 Fayetteville Street,
Durham, NC 27707
919-530-6297
ssendlin@nccu.edu

Clyde R. Metz
College of Charleston
Department of Chemistry and Biochemistry
66 George Street,
Charleston, SC 29424
843-953-8097
metzc@cofc.edu

ABSTRACT
In this paper we describe an ongoing project where the goal is to develop competence and confidence among chemistry faculty so they are able to utilize computational chemistry as an effective teaching tool. Advances in hardware and software have made research-grade tools readily available to the academic community. Training is required so that faculty can take full advantage of this technology, begin to transform the educational landscape, and attract more students to the study of science.

Categories and Subject Descriptors
J.2 [**Physical Sciences and Engineering**]: Chemistry

General Terms
Algorithms, Design, Experimentation, Theory, Verification.

Keywords
Computational Chemistry Education, Workshops, Graduate, Undergraduate.

1. INTRODUCTION
The majority of today's students are technologically savvy and are often more comfortable using computers than the faculty who teach them. In order to harness the student's interest in technology and begin to use it as an educational tool, most faculty members require some level of instruction and training. Because chemistry research increasingly utilizes computation as an important tool, our approach to chemistry education should reflect this. The ability of computer technology to visualize and manipulate objects on an atomic scale can be a powerful tool to increase both student interest in chemistry as well as their level of understanding. Computational Chemistry for Chemistry Educators (CCCE) is a project that seeks to provide faculty the necessary knowledge, experience, and technology access so that they can begin to design and incorporate computational approaches in the courses they teach. We also work from the viewpoint that technology should *enhance* the educational experience and allow teaching of concepts in new and more effective ways. In other words, just because something *can* be done computationally does not necessarily mean that it *should* be.

This project takes a broad view of what computational chemistry entails. In addition to molecular modeling, the general use of computer technology to demonstrate and help students learn chemical principles is included. This includes visualization, animation, data manipulation and graphing, system dynamics software, computer algebra systems, etc.

2. DESCRIPTION
Initial planning for a one week workshop that would provide hands-on instruction in the use of molecular modeling software began in 2002 with support from the Shodor Education Foundation through the National Computational Science Institute (NCSI) program, a project supported in large part by the National Science Foundation (NSF).[1] Subsequent support has been provided by the Supercomputing (SC) Education Program[2] and most recently by the NSF-supported Center for Workshops in the Chemical Sciences (CWCS) project.[3]

The CCCE workshop was the first in a series of NCSI discipline-specific computational science workshops, and provided a template for those that followed.[4] The topics chosen for the molecular modeling workshop were:

1. Introduction to Computational Chemistry
2. Basis Sets
3. Choice of Theoretical Method

4. Single Point Energies and Geometry Optimization
5. Electron Densities, Electrostatic Potentials, and Reactivity Predictions
6. Modeling in Solution
7. Computing Spectroscopic and Thermochemical Properties
8. Quantitative Structure Activity/Property Relationships
9. Transition States
10. Computational Study of System Dynamics (Chemical Kinetics)
11. Biochemical Applications

Each session consists of a lecture followed by hands-on computer exercises that illustrate the lecture concepts.[5] These sessions provide the participants with sufficient breadth and depth so that by the end of the week they are able to choose an appropriate theoretical model chemistry and basis set, and also know how to set up, perform, and interpret calculations. Participants are not expected to become expert quantum chemists! This is not the goal of the workshop. We seek to provide faculty with a level of molecular modeling knowledge so that they can competently use the tool and have confidence in the results of their calculations. In particular, we hope that the participants are able to answer the following questions after the workshop:

a) What do I want to know about a molecular or ionic system?
b) How accurately do I need to know it?
c) How long am I willing to wait for the answer to be computed?
d) What software/hardware can I use to accomplish the task?

The answers to the above questions then dictate the possible computational approaches that should be used. The level of knowledge the participants should gain is similar to that of many modern spectroscopic techniques that are regularly employed by chemists. One does not need to be an expert in the mathematics and quantum mechanics involved in nuclear magnetic resonance (NMR) spectroscopy to be able to correctly interpret the results of NMR experiments. However, we believe that the user should also be beyond the "black box" level of understanding, and have a firm grasp of what a given technique can or cannot do. We believe that the topics are addressed in sufficient detail for the participants to attain the desired level of expertise.

In addition to the molecular modeling-focused topics, a session on chemical kinetics is included. Spreadsheet-based activities are incorporated, but the main hands-on portion of the exercise is based on systems dynamics software[6] with which many faculty members are unfamiliar. We also mention computer algebra systems (Mathcad, Matlab, Mathematica, etc.) and include example code so that those with access to one of these programs could begin to build their own chemical kinetics model.

Other non-molecular modeling sessions focus on free or low-cost software, browser plug-ins such as molecular viewers, and online repositories of information that have proven useful for educational purposes. These sessions are either hands-on or demonstrations that explore the use of each resource. Examples include:

Free molecular drawing programs
ACD ChemSketch/3D: http://www.acdlabs.com/download/
BioRad KnowItAll: http://www.knowitall.com/academic/

Web-Based Resource Collections
Computational Science Education Reference Desk (CSERD): http://www.shodor.org/refdesk/
Interactivate: http://www.shodor.org/interactivate/
Virtual Lab Simulator: http://www.chemcollective.org/vlab/vlab.php
Netlogo: http://ccl.northwestern.edu/netlogo/
ReciprocalNet: http://www.reciprocalnet.org/
Protein Data Bank: http://www.rcsb.org/pdb/home/home.do
Journal of Chemical Education: http://jchemed.chem.wisc.edu/
Proteopedia: http://www.proteopedia.org

Viewers for molecular structure
CHIME: http://www.symyx.com/support/developer/chime/index.jsp
VRML: http://www.parallelgraphics.com/products/cortona3d/
Jmol: http://jmol.sourceforge.net/
Mercury: http://www.ccdc.cam.ac.uk/free_services/mercury/

Freely available Graphical User Interfaces, computational engines, and Viewers for molecular modeling
RUNpcg: http://www.chemsoft.ch/qc/RUNpcg.htm
Firefly: http://classic.chem.msu.su/gran/gamess/
Molekel: http://molekel.cscs.ch/wiki/pmwiki.php

Workshop participants apply their new knowledge as they begin to develop a case study that they will use in a course that they regularly teach. Time to work on the case study is included in the daily workshop schedule. The workshop culminates with presentations of these case studies on the final day. Some of the developed case studies are appropriate for use during several minutes of lecture time, while others are designed to occupy several hours during a laboratory period. Many of these projects focus on the use of computation to teach old topics in a new (and hopefully, more effective) manner. Others introduce students to molecular modeling and instruct students on how to build molecules, perform calculations, and interpret their own results. Some are designed as homework assignments that students complete outside of class on their own time. One very effective educational approach is to use a series of computations to provide results that the students then must interpret, identify trends, and make sense of the trend using the knowledge of chemistry that they have gained to that point. This discovery-mode of learning can help students to understand and retain the information more effectively.[7]

Other week-long workshops have been developed and taught at both the Introductory and Advanced levels. Introductory workshops place less emphasis on molecular modeling and focus more on the other topics mentioned. The Advanced workshops cover molecular modeling subjects of interest to the attendees in more detail and provide additional time for collaboration among the attendees to develop classroom materials.

A number of shorter workshops have also been provided. These range from half-day sessions at local and regional ACS meetings[8] or Supercomputing Education Program events[9] to full- or multiple-day sessions arranged with individual institutions.

3. RESULTS

3.1 Workshops

Since 2002, the week-long molecular modeling workshop has been taught eight times with ~165 attendees. Advanced and Introductory workshops have been taught twice each with a total of ~60 participants. Thirteen shorter workshops that cover the

basics of molecular modeling and other topics have also been held with over 200 participants attending. Computational chemistry has also been included as a major part of an ongoing high school teacher development project in Illinois with over 120 participants.[10] Over the past seven years, the CCCE project has thus reached well over 500 college faculty and high school teachers. We hope to continue our work and further increase these numbers.

Workshop participants and instructors are active in presenting their work at various professional meetings[11] and in appropriate journals.[12] This is an expected outcome of workshop attendance and we hope that the current journal will become a popular venue for sharing successful activities.

Teaching these workshops has led the authors to initiate new courses at their own institutions. The College of Charleston has both Introduction to Modeling in Chemistry and Advanced Physical Chemistry: Molecular Modeling courses. North Carolina Central University has an interdisciplinary, team-taught Introduction to Computational Science and Informatics course as well as a graduate-level Computational Chemistry course. Computational content has also been added in other courses where appropriate.

3.2 Workshop Assessment

Each workshop was evaluated using three different types of anonymous on-line evaluation forms. A pre-workshop survey identified the level of current knowledge and available resources before the workshop began. Daily evaluations helped the instructors determine participant progress and provided session-specific feedback. A post-workshop survey evaluated overall advancement in knowledge and gave an indication of the level of competence and confidence in using computational tools that the participants had gained.

The post-workshop survey consisted of a series of questions to which the participant could respond with simple "yes or no" answers, rate the response over a numerical scale, and generate free-style short responses. The following example questions and responses were chosen from 25 completed surveys by college faculty for the 2008 and 2009 molecular modeling workshops. To simplify the presentation of results, numerical scores were assigned to the various responses as indicated below and the average score is given with the standard deviation noted in parentheses.

Scoring for questions 1-3: [strongly disagree = 0, somewhat disagree = 1, somewhat agree = 2, strongly agree = 3]

1. Did the workshop advance the participant's knowledge of computational science? 2.56 (0.90)

2. Did the participant learn to use computational science in teaching? 2.52 (0.85)

3. Did the participant learn to use models applicable to chemistry? 2.36 (0.84)

4. Did the participant learn to use models to demonstrate a concept in a course? *Affirmative* for 96% of the responses

5. Will the participant work with individual instructors at the home institution to assist them to learn more about computational science education activities? *Affirmative* for 88% of the responses

6. The self-rated confidence level on using computational science to enhance the learning experience of the students [not at all = 1, very confident = 10] of the participants was 8.24 (1.63)

The above results combined with verbal and e-mail comments made by participants during and following the workshops indicate substantial gains in both knowledge and confidence for using computational approaches in the classroom.

3.3 Related Accomplishments

The authors have also become involved in several related projects. UNChem is an online chemistry fundamentals review course[13] designed for students who will begin freshman chemistry. The site is being reviewed and updated. Many of the resources used in the CCCE workshops will be incorporated into the materials. The Computational Science Education Reference Desk (CSERD),[14] a Pathways project of the National Science Digital Library[15] and funded by the NSF, aims to help students learn about computational science and to help teachers incorporate it into the classroom. The authors and workshop attendees have edited and reviewed much of the chemistry content of CSERD.

A very useful workshop product has been the CCCE web site available at: http://www.computationalscience.org/ccce/. The site currently houses the molecular modeling workshop lecture and laboratory materials, a reference book list, a link to a glossary of computational chemistry terms, and links to other chemistry resources. Site visitors most often utilize the Labs link (shown on the next page) which has a matrix of ten molecular modeling topics that have been translated into exercises that can be performed with seven different popular molecular modeling programs. The exercises contain scaffolded instructions which, when followed in sequence, teach the user how to effectively use the software package and to also gain experience with different computational methods. Also included are instructions in the use of spreadsheets, computer algebra software, and systems dynamics programs for the modeling of chemical kinetics.

In the fall of 2009, Computational Chemistry for Chemistry Educators was awarded the Undergraduate Computational Engineering and Science (UCES) Award. This award program was created to promote and enhance undergraduate education in computational engineering and science and encourages further development of innovative educational resources and programs, recognizes the achievements of CES educators, and disseminates educational material and ideas to the broad scientific and engineering undergraduate community. The UCES Awards Program is funded by the Department of Energy and administered by the Krell Institute.[16]

3.4 Technology

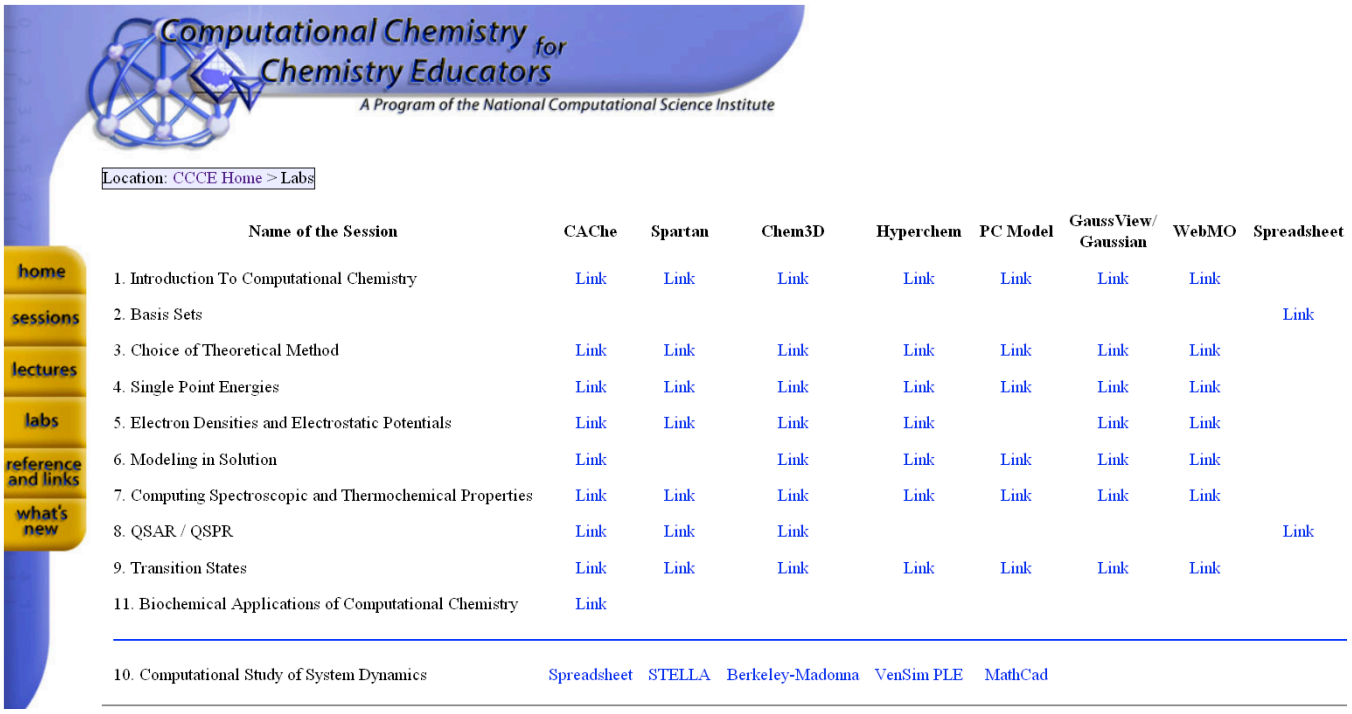
The first several years of the CCCE program used PC-based molecular modeling software, such as CAChe,[17] Spartan,[18] Hyperchem,[19] or PC Model[20] for all hands-on exercises. This approach necessitated loading (and troubleshooting) many programs and license files on multiple machines. In order for participants to have software to work with when they returned to their home institution, a mini-grant program was used where participants would apply for funds to cover the cost of a software package, a computer RAM or HD upgrade, plus a few books. In 2006 we began to use WebMO[21] which is a server-based GUI that

allows drawing molecules, setting up and submitting jobs, and visualizing the results. The WebMO software interfaces to computational engines such as MOPAC,[22] Gaussian,[23] Tinker,[24] and NWChem.[25] The switch to a server-based system greatly reduced the individual computer set-up time and also allows participants to have continued access to their files following the workshop. This thin client model also makes it easy to provide the students of our workshop participants with access to the software as well. It is, after all, the students and their learning outcomes that we want to promote.

4. FUTURE WORK

As individual software packages undergo periodic updates, the exercise instructions provided on the CCCE site must be changed. A new and improved version of the site is being developed and will include new instructions where needed, along with desktop-capture video instructions on using different software programs. We also hope to replace the current, static lecture materials with information that is more interactive and includes some video and voiceover.

For the past eight years, a main thrust of CCCE has been focused on electronic structure calculations for single molecules. This focus is largely a result of the hardware and software that has been readily available. Our move to the use of server-based WebMO removes some of the hardware constraints and allows us to begin planning for the inclusion of more reaction dynamics and molecular dynamics calculations and exercises. The workshops currently use the Odyssey software package[26] for molecular dynamics exercises. While this package is quite useful for educational purposes, it is a commercial product and many workshop attendees cannot afford it. Researchers use molecular dynamics programs such as freely available GROMACS[27] and NAMD.[28] To extend the CCCE model in this direction will require a GUI, several of which are available,[29] and hardware on which to perform the simulations. Work to provide these resources in ongoing. As ever-larger computing resources come online, such as the Blue Waters petascale machine,[30] CCCE hopes to assist faculty and teachers in learning how to use these systems in a scientifically sound and educationally productive manner. It is incumbent upon the educators of today to motivate and train our students to become the scientists of tomorrow. Computational modeling can help interest more students in science and also provide the skills our graduates will need to harness the power of future computer technology.



5. ACKNOWLEDGEMENTS

The authors wish to thank Dr. Robert M. Panoff, Executive Director and President of the Shodor Education Foundation for his years of support and encouragement, Mr. Bob Gotwals of the North Carolina School for Science and Mathematics for his assistance in the initial workshop planning stages, Dr. Elizabeth Bell-Loncella of the University of Pittsburgh at Johnstown for recently joining the instructional team, and Dr. Brad Stone of San Jose State University for having hosted four CCCE workshops. Dr. Jay Mashl of the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and Charlie Peck of Earlham College have provided invaluable software and hardware support. We would also like to thank the other workshop hosts as well as the funding agencies whose support made this work possible: NSF DUE 0127488, SC Education Program, and Dr. Jerry Smith with the Center for Workshops in the Chemical Sciences (NSF CCLI-ND).

6. REFERENCES

[1] National Computational Science Institute: http://computationalscience.org/.

[2] SC Education Program: http://sc10.sc-education.org/.

[3] Center for Workshops in the Chemical Sciences: http://chemistry.gsu.edu/CWCS/.

[4] Other workshops are available in Biology, Computational Thinking, Parallel and Cluster Computing, and Physics. See: http://www.computationalscience.org/workshops/.

[5] For additional details, see the CCCE web site at: http://www.computationalscience.org/ccce/.

[6] http://www.vensim.com/sdmail/sdsoft.html.

[7] Altintas, I.; Berkley, C.; Jaeger, E.; Jones, M.; Ludascher, B.; Mock, S., "Kepler: An Extensible System for Design and Execution of Scientific Workflows," *Proceedings of the 16th Int'l Conf. Scientific and Statistical Database Management,* IEEE Press, **2004**, pp. 423-424.

[8] For college and university faculty: http://acs.confex.com/acs/56serm/techprogram/S261.HTM.

For high school teachers: http://acs.confex.com/acs/56serm/techprogram/S953.HTM

[9] http://moodle.sc-education.org/.

[10] Institute for Chemistry Literacy through Computational Science (ICLCS): http://iclcs.uiuc.edu/.

[11] Symposia: http://acs.confex.com/acs/56serm/techprogram/S281.HTM. Faculty workshops: http://acs.confex.com/acs/56serm/techprogram/S261.HTM. Teacher workshops: http://acs.confex.com/acs/56serm/techprogram/S953.HTM.

[12] (a) Sendlinger, S.C. and Metz, C.R., "CSERD: Another Important NSDL Pathway for Computational Chemistry Education", *J. Chem. Ed.*, **2009**, *86*, p. 126. (b) Sendlinger, S.C.; DeCoste, D.J.; Dunning, T.H.; Dummitt, D.A.; Jakobsson, E.; Mattson, D.R.; Wiziecki, E.N., "Transforming Chemistry Education through Computational Science", *Computing in Science and Engineering*, **2008**, *10(5)*, pp. 34-39.

[13] http://www.shodor.org/unchem/.

[14] http://www.shodor.org/refdesk/.

[15] http://nsdl.org/.

[16] http://www.krellinst.org/csgf/awards-contests/uces-award-program

[17] http://solutions.us.fujitsu.com/www/content/news/newsdetail.php?nf=07230560.nitf

[18] http://www.wavefun.com/products/spartan.html.

[19] http://www.hyper.com/.

[20] http://www.serenasoft.com/.

[21] http://www.webmo.net/.

[22] http://openmopac.net/home.html.

[23] http://www.gaussian.com/.

[24] http://dasher.wustl.edu/tinker/.

[25] http://www.nwchem-sw.org/index.php/Main_Page.

[26] http://www.wavefun.com/products/odyssey/odyssey.html.

[27] http://www.gromacs.org/.

]28] http://www.ks.uiuc.edu/Research/namd/.

[29] (a) Roopra, S.; Knapp, B; Omasits, U.; Schreiner, W., "jSimMacs for GROMACS: A Java Application for Advanced Molecular Dynamics Simulations with Remote Access Capability", *J. Chem. Inf. Model.*, **2009**, *49(10)*, pp. 2412-2417. (b) Sellis, D.; Vlachakis, D.; Vlassi, M., "Gromita: A Fully Integrated Graphical User Interface to Gromacs 4", *Bioinformatics and Biology Insights*, **2009**, *3*, pp. 99-102. (c) http://kde-apps.org/content/show.php/%20Gromacs+GUI+?content=47665.

[30] http://www.ncsa.illinois.edu/BlueWaters/.

Testing the Waters with Undergraduates (If you lead students to HPC, they will drink)

Angela B. Shiflet
Department of Computer Science
Wofford College
Spartanburg, S. C. 29303 USA
001-864-597-4528
shifletab@wofford.edu

George W. Shiflet
Department of Biology
Wofford College
Spartanburg, S. C. 29303 USA
001-864-597-4625
shifletgw@wofford.edu

ABSTRACT
For the Blue Waters Undergraduate Petascale Education Program (NSF), we developed two computational science modules, "Biofilms: United They Stand, Divided They Colonize" and "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better." This paper describes the modules and details our experiences using them in three courses during the 2009-2010 academic year at Wofford College. These courses, from three programs, included students from several majors: biology, chemistry, computer science, mathematics, physics, and undecided. Each course was evaluated by the students and instructors, and many of their suggestions have already been incorporated into the modules.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer Science Education, Curriculum

General Terms
Design, Experimentation, Measurement.

Keywords
Computational Science, High-Performance Computing, Educational Modules, Biofilms, Social Networks, Blue Waters, Undergraduate, Petascale.

1. INTRODUCTION
With NSF funding, the Blue Waters Undergraduate Petascale Education Program [1] is helping to prepare students and teachers to utilize high performance computing (HPC), particularly petascale computing, in computational science and engineering (CSE). UPEP supports three initiatives:

- *Professional Development Workshops* for undergraduate faculty
- *Research Experiences* for undergraduates
- *Materials Development* by undergraduate faculty for undergraduates

The goal of the Materials Development initiative is "to support undergraduate faculty in preparing a diverse community of students for petascale computing."

For this program, the authors developed and class tested two computational science modules, "Biofilms: United They Stand, Divided They Colonize" and "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better," which are available on the UPEP Curriculum Modules site [2]. This paper describes and discusses the modules and our experiences using them in the courses Modeling and Simulation, High Performance Computing, and Mathematical Modeling at Wofford College [3] during the 2009-2010 academic year.

Several of the students in these classes are obtaining Wofford's Emphasis in Computational Science (ECS). Bachelor of Science students may obtain an ECS by taking Calculus I, Introduction to Programming and Problem Solving (in Python), Data Structures (in Python and C++), Modeling and Simulation, and Data and Visualization and doing a summer internship involving computation in the sciences [4]. Meaningful applications that illustrate fundamental concepts and techniques, such as those in the above modules, are crucial in their computational science education.

2. MODULES
2.1 Pedagogy
Prerequisites are for the modules are minimal, requiring no programming or calculus background but the maturity to read the material. Students who used the modules ranged from first- to fourth-year with majors from biology, chemistry, physics, mathematics, computer science, and undecided. Both modules provide the biological background necessary to understand the applications, the mathematical background needed to develop

models, and references for further study. "Biofilms: United They Stand, Divided They Colonize" has ten (10) multi-part quick review questions with answers to provide immediate feedback, while "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better" has six (6) such questions. The former module provides twenty-three (23) project assignments, and the latter has nineteen (19) projects for further exploration.

To help with implementation, example solutions in various systems are available for download from the UPEP Curriculum Modules site [1]. Accompanying the biofilms module are a *Mathematica* implementation and high performance computing simulations for two and multiple processors in C/MPI along with an accompanying walkthrough of the code by student intern Shay M. Ellison. The epidemic module has simulations in *Mathematica* for a small dataset of 18 people and for a large dataset of 1000 people randomly selected from Network Dynamics and Science Simulation Laboratory (NDSL) synthetic data for the population of Portland, Oregon [5].

2.2 High Performance Computing in Modules

In line with the aims of UPEP, both modules have a section on "Computing Power" that discusses the need for high performance computing (HPC) within the contexts of the particular applications. The biofilms section on the topic points out that biofilms are highly complex with numerous features and the version in the module considers form, not function, in 2D as opposed to 3D. Thus, HPC is usually necessary for more involved and realistic biofilms models. The other module discusses how petascale computing is needed for processing large datasets involving millions of people and their activities and for more sophisticated computations, such as studying the nature of epidemics and the impacts of policy decisions on controlling epidemics in urban environments.

With each module, the "Computing Power" section can be covered for information only, as a starting point for class discussion, or as motivation for the students' own HPC project development. Moreover, students can develop sequential or high performance computing versions of many of the projects with some assignments requiring HPC and others asking for timing comparisons of parallel codes and their sequential counterparts.

2.3 Class Testing

At the end of the semester, students in Mathematical Modeling, which used the epidemics module, and High Performance Computing, which used both modules, were asked to complete a questionnaire about the modules, while students in Modeling and Simulation completed a general questionnaire about the course. The first author taught the latter two courses, while another professor taught Mathematical Modeling.

The questionnaires for the HPC class had the students rate the following statements from 1 (strongly disagree) to 5 (strongly agree):

- I understood the science applications in the module.
- I understood the mathematics in the module.
- I understood the algorithms in the module.
- The module was readable.
- The program helped me understand parallel processing with MPI.
- My team/I could complete the C/MPI program.

They were also asked to elaborate about the above scores, particularly those below 4, to indicate what they like best and what they found most difficult about the module, to give corrections and suggestions for improvement, and to list under what circumstances would they anticipate that high performance computing would be useful in modeling that application.

The questionnaire for the Mathematical Modeling class, which did not use HPC, included the first four questions above along with the following 1-5 rated questions:

- The modeling assignment(s) helped me understand the material.
- My team/I could complete the modeling assignment(s).

They were asked similar discussion questions to those for the HPC class as well as the modeling project(s) they did related to this module and under what circumstances would they anticipate that high performance computing would be useful in modeling social networks. The professor completed a similar questionnaire and responded to a follow-up email about the module. Unfortunately, because the questionnaire was distributed at the end of the semester, only one Mathematical Modeling student and the professor completed the questionnaire.

Results of the questionnaires are described below.

3. MODULE 1: BIOFILMS

3.1 Scientific Question

A biofilm is a community of very small organisms that adhere to a surface (substratum) in an aqueous environment [6]. Examples of this ubiquitous phenomenon are dental plaque, the sticky substance covering the breathing passages of cystic fibrosis patients, antibiotic resistant bacterial colonies, and the microbial film in wastewater treatment. Because these communities have such important impacts, scientists are seeking to understand better the structure and function of biofilms [7], and the application provides good motivation for computational science students in courses such as High Performance Computing and Modeling and Simulation.

3.2 Computational Models and Algorithms

The module developed a cellular automaton simulation in two dimensions (2D) of the formation of the structure of a biofilm without regard to its function. Each discrete time step of the simulation contained the following phases:

- Diffusion of nutrients
- Growth and death of microbes
- Consumption of nutrients by microbes

The module covers the basics of cellular automaton simulations including boundary conditions and models for diffusion, biofilm growth model, and nutrient consumption. Afterwards, the material develops generic algorithms for the models, a simulation program, and a visualization.

3.3 Assessment of Simulation's Results

A series of module figures shows several steps of a visualization of bacteria grids and associated nutrient grids, such as in Figure 1 of this article. Using the figures, a section on "Rubric for Assessment" examines empirically successes and shortcomings of this simulation of a structural formation of a biofilm.

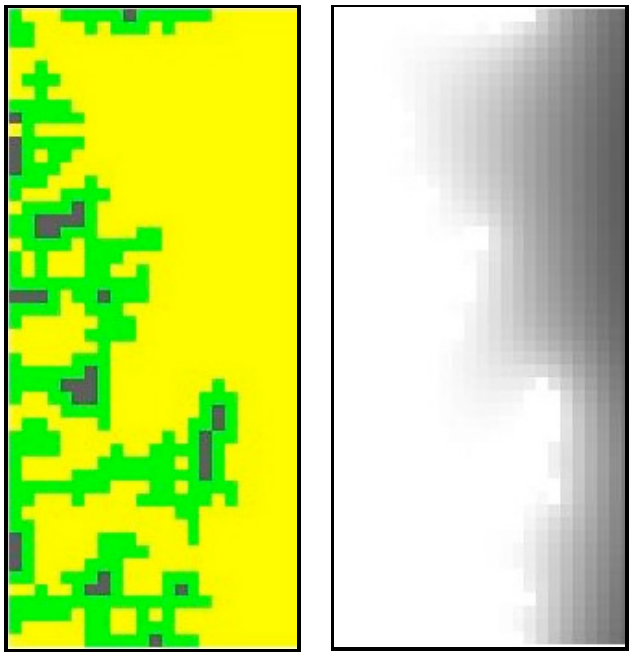


Figure 1. Simulated biofilm with corresponding nutrient grid

3.4 Class Testing in Modeling and Simulation

With *Introduction to Computational Science: Modeling and Simulation for the Sciences* [8] as a text, half of Wofford's Modeling and Simulation course (COSC/MATH 201) covers system dynamics modeling using a tool, such as *STELLA*®, *Vensim*®, or *Berkeley Madonna*®. The second half of the semester employs a computational tool, such as *Mathematica*®, *Maple*®, or *MATLAB*®, for developing cellular automata (CA) simulations, where the world under consideration consists of a rectangular grid of cells and each cell has a state that can change with time according to rules. We start the second part of the semester by considering random walk simulations and having projects on the formation of polymers and crystals. Additional important CA algorithms involve diffusion and spreading, and simulation of the formation of biofilm structures provides a significant application of these techniques.

With the background provided by this application and these concepts, over a two-and-a-half week period, the four students in the class (three biology majors and a mathematics major, all ECS students; a freshman, a junior, and two seniors) were able to revise the diffusion program, which was considered in detail in the class; in pairs to develop additional computational science applications, such as modeling the growth of mushroom fairy rings and modeling animal use of cognitive maps to find food; and to perform well on quiz and test questions on the material.

Moreover, at the end of the term, students had a brief introduction to concurrent processing and parallel algorithms. Consideration of more complex and larger biofilm arrangements, particularly in 3D, illustrated the importance of high performance computing in computational science.

3.5 Class Testing in High Performance Computing Course

The High Performance Computing course (COSC 365) at Wofford was populated by a mixture of Emphasis in Computational Science (ECS) students and computer science majors (five students: one biology/ECS, one chemistry/ECS, one computer science/chemistry/ECS, two computer science; sophomore to senior level). All students had had through Data Structures with programming in Python and C++ and at least one other computer science or computational science course.

With a week of class time devoted to the biofilms module, the class in teams successfully developed MPI/C programs for diffusion using a parallel random number generator on NCSA's Teragrid computer Abe, a Dell Intel 64 Linux Cluster [9]. Later, the students also demonstrated their understanding of the material with their performance on a test and an exam.

In the course, the emphasis is on learning HPC techniques, and diffusion is important in many applications that require high performance computing. Biofilms were particularly interesting to the biology and chemistry students in the class. Moreover, the topic helped computer science majors make the connections of theory to application, which they sometimes overlook, and seemed to provide motivation for all the students. For example, evaluation comments indicated that the module is "very easy to read," and the model "very useful for the real world problems."

3.6 Blue Waters UPEP Internship Involvement

During the summer of 2009, student Shay Ellison had a Blue Waters UPEP Internship to develop parallel versions for two and multiprocessors of the biofilms application and to write a ten-page accompanying tutorial, "Biofilms Parallel Computational Model with MPI and C," which are also available on the NCSI UPEP Curriculum Modules site [2]. Using TeraGrid resources, he investigated parallel random number generation, output of results, and speedup of the program with multiple processors. The experience enhanced his understanding of HPC and undoubtedly is valuable to Shay as he pursues graduate studies in information security at Florida State University.

3.7 Additional Outreach

In an effort to extend the educational benefits of the module, the authors wrote and presented a paper, "Simulating the Formation of Biofilms in an Undergraduate Modeling Course" for the Workshop on Teaching Computational Science at the International Conference on Computational Science (ICCS) in Amsterdam [10].

4. MODULE 2: SOCIAL NETWORKS

4.1 Scientific Question

The module "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better," which has the potential of making similar impacts, covers social networks and individual-based epidemiology simulations. Individual-based (or network-based) epidemiology simulations that track the simulated behavior of individuals in a community provide greater specificity and are easier to verify than cellular automaton simulations [11]. The module discussed the following important metrics for social networks:

- a smallest set of locations (minimum dominating set) that a given proportion of the population visits, which can be helpful in determining sites for fever sensors or in closing of particular public buildings during an epidemic

- the distribution of the number of contacts people have with other people (degree distribution), which can facilitate targeted vaccination of individuals who have many contacts [12]
- the probability that two contacts of a randomly chosen person have contact with one another (clustering coefficient), which is an indication of how rapidly a disease can spread through a community [13]
- the average smallest number of contacts for a disease to spread from one arbitrary individual to another (mean shortest path length), which also indicates the rapidity with which a disease can spread

4.2 Computational Models

The basic data structure for solving this scientific problem is a graph, or a set of nodes with undirected or directed edges connecting some of the points. For a contact network or a social network, the nodes represent people or groups of people, such as members of a household that can become infected, and places, where the disease can spread from an infected person to a susceptible individual (See Figure 2). Each edge represents an association that can lead to transmission of the disease. Thus, besides the biological background, the module covers some of the fundamental concepts in graph theory, such as adjacent nodes, degree, complete graph, and paths. Moreover, students explore some of the characteristics of such social networks and of biological networks in general, such as the following:

- Social networks are scale-free, where most nodes have relatively low degree but a few nodes, called hubs, have high degrees, making such networks particularly vulnerable to attack and failure [12].
- Biological networks exhibit the small world property, where the average length of a path between nodes is small in comparison to the size of the graph, so that these graphs are efficient communicators of information or disease.

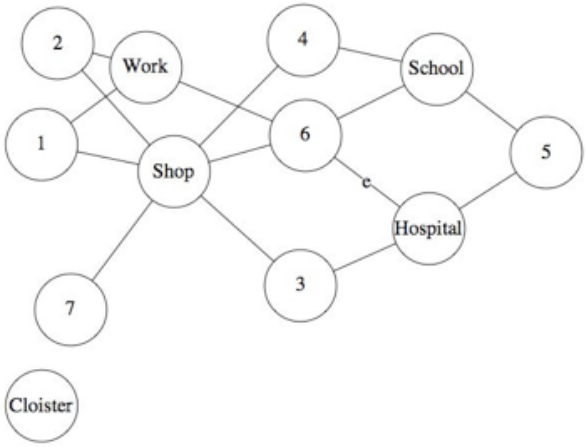


Figure 2. Contact network of households and places

4.3 Algorithms

For implementation of the graph data structure, the module employs a vector to store nodal values, adjacency matrices to represent edge connections and associated values, and connection matrices for existence of associations only. With data of people's activities, we can construct a people-location graph. The module covers the FastGreedy Algorithm to obtain an approximation of a minimum dominating set, or a smallest set of locations that a given proportion of the population visits. Forming a people-people graph of contacts that individuals have with each other by visiting the same locations in a day, we can also compute a distribution of the number of contacts people have with others and the clustering coefficient of each person.

4.4 Assessment of Models

The module and one accompanying *Mathematica* file showed computation of various metrics for 1000 people randomly selected from NDSSL's synthetic data for the population of Portland, Oregon [5]. For example, the degree distribution of this data in Figure 3 approximates the power law, $P(k) = ck^{-r}$, common for scale-free networks. The shape reveals only a few critical nodes have degree 6 or more. The module points out that using further demographic information, public health officials might target such people for immediate vaccination.

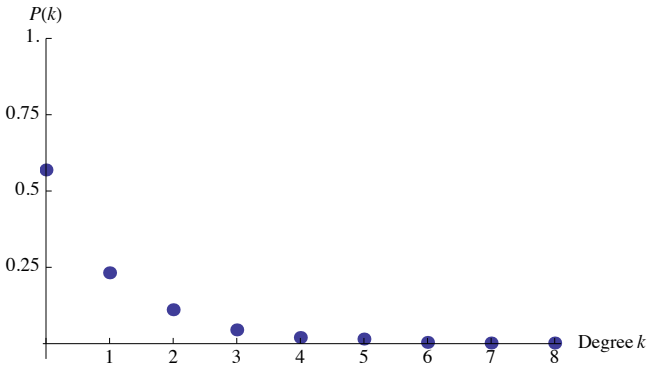


Figure 3. Degree distribution of 1000 randomly selected people

4.5 Need for High Performance Computing

NDSSL's synthetic data for the population of Portland, Oregon involves 1,615,860 people having 8,922,359 activities [5]. Using such data, computational scientists are developing high performance individual-based simulation models to study the nature of epidemics and the impacts of policy decisions on controlling epidemics in urban environments. Because such individual-based simulations incorporate massive amounts of data that require extensive effort to gather and need massive computing power to process, such models provide excellent examples for HPC for the students.

4.6 Class Testing in Mathematical Modeling

During spring 2010, the Mathematical Modeling course at Wofford used this module with *Mathematica* for six class hours, half of which was formal instruction and half hands-on activity in which the students worked through the "base model." Then, deriving their own modifications, students described the physical system and how it was incorporated into the model, solved the model with *Mathematica*, and interpreted the results. Students' learning on this topic was assessed through participation in class activities, journal entries about the reading, and the above assignment.

Class members, typically at the junior-senior level, were two physics, one mathematics/physics, one mathematics/chemistry,

two mathematics majors, and one undecided. The professor, Dr. Anne Catlla, indicated that with background of the module and accompanying files students were able to successfully complete the assignments and that overall there was more variation among individuals than among majors.

A student in the class stated, "I liked the application of social network theory and graph theory to a disease modeling scenario." The professor wrote she "thought that the assignments (both the Quick Review problems and the projects at the end of the module) were excellent;" and in an evaluation she "strongly agreed" that she understood the science applications, mathematics, and algorithms in the module. Although the class did not use HPC, coverage of the material helped to enlighten them on the need for and advantages of using such power for larger data sets and problems.

4.7 Class Testing in High Performance Computing

Over a three-week period at the end of the spring 2010 semester, students in High Performance Computing at Wofford heard from the professor about important graph theory applications and techniques in HPC, studied the epidemics module, presented the material to the rest of the class, and in two teams developed C/MPI programs to calculate various metrics discussed above. The teams implemented their solutions on the Teragrid's NICS Kracken, a 99072-processor Cray XT5 computer [9]. Later, they completed exam questions on the material. All students did well on the module, although the student with the least amount of programming background experienced more difficulty with the HPC aspects of the assignments.

In an evaluation, one student commented, "If detailed enough information was provided and the data set was realistically large, HPC would be invaluable in modeling social networks." Another stated, "I liked being able to see the relation to science in this module." A third student wrote, "I particularly liked the use of a two-dimensional array as a connection graph. We had not really used matrices in such a way before." One of the students was delighted to relate the application to her own research involving the electrical grid in a summer internship at Oak Ridge National Laboratory.

5. CONCLUSION

The limited class sizes preclude statistical analysis at this time, but we are encouraged from the very positive responses from students and professors, alike. We have already incorporated suggestions from participants in all three classes to make the modules more effective. Blue Waters funding in the Undergraduate Petascale Education Program has been invaluable in high performance computing computational science module development, internships, conference participation, teaching, and learning. This project is advancing education personally, locally, nationally, and internationally.

6. REFERENCES

[1] National Computational Science Institute Undergraduate Petascale Education Program (UPEP). http://computationalscience.org/upep

[2] National Computational Science Institute Undergraduate Petascale Education Program (UPEP) Curriculum Modules, UPEP Curriculum Modules site. http://computationalscience.org/upep/curriculum

[3] Wofford College. http://www.wofford.edu

[4] Computational Science - Wofford College. http://www.wofford.edu/computationalscience/

[5] NDSSL (Network Dynamics and Simulation Science Laboratory, Virginia Polytechnic Institute and State University). 2009. "NDSSL Proto-Entities" http://ndssl.vbi.vt.edu/opendata/.

[6] Donlan, R. M. and Costerton, J. W. 2002. Biofilms: survival mechanisms of clinically relevant microorganisms. Clin. Microbiol. Rev. 15 (2) 167-193 ().

[7] Stewart, Philip S. 2003. Guest Commentaries, Diffusion in Biofilms. In *J. Bacteriol*. 185(5): 1485–1491.

[8] Shiflet, A. and Shiflet, G. 2006. *Introduction to Computational Science: Modeling and Simulation for the Sciences*. Princeton University Press, Princeton.

[9] Teragrid. 2010. https://www.teragrid.org/

[10] Shiflet, A. and Shiflet, G. 2010. Simulating the Formation of Biofilms in an Undergraduate Modeling Course ICCS 2010. In *Procedia Computer Science*. Elsevier. 1(1): 895-901.

[11] Bisset, Keith and Marathe, Madhav. 2009. "A Cyber Environment to Support Pandemic Planning and Response." *SciDAC Review* 13: 36-47. http://www.scidacreview.org/0903/html/maranthe.html.

[12] Mason, Oliver and Verwoerd, Mark. 2007. Graph Theory and Networks in Biology. In *IET Systems Biology* 1: 89-119. http://www.hamilton.ie/SystemsBiology/files/2006/graph_theory_and_networks_in_biology.pdf.

[13] Newman, M. E. J., Watts, D. J., and Strogatz, S. H. 2002. Random graph models of social networks *Proceedings of the National Academy of Science* 99 (Suppl 1): 2566-2572. http://www.pnas.org/content/99/suppl.1/2566.full.

Parallelization of Particle-Particle, Particle-Mesh Method within N-Body Simulation

Nicholas W. Nocito
Kean University, NJCSTM
1000 Morris Avenue
Union, NJ 07083
nociton@kean.edu

ABSTRACT
The N-Body problem has become an intricate part of the computational sciences, and there has been rise to many methods to solve and approximate the problem. The solution potentially requires on the order of N^2 calculations each time step, therefore efficient performance of these N-Body algorithms is very significant [5]. This work describes the parallelization and optimization of the Particle-Particle, Particle-Mesh (P3M) algorithm within GalaxSeeHPC, an open-source N-Body Simulation code. Upon successful profiling, MPI (Message Passing Interface) routines were implemented into the population of the density grid in the P3M method in GalaxSeeHPC. Each problem size recorded different results, and for a problem set dealing with 10,000 celestial bodies, speedups up to 10x were achieved. However, in accordance to Amdahl's Law, maximum speedups for the code should have been closer to 16x. In order to achieve maximum optimization, additional research is needed and parallelization of the Fourier Transform routines could prove to be rewarding. In conclusion, the GalaxSeeHPC Simulation was successfully parallelized and obtained very respectable results, while further optimization remains possible.

General Terms
Performance, algorithms, design

Keywords
N-Body, Message Passing Interface, Particle-Mesh, Fourier Transform

1. INTRODUCTION
The N-body problem is an example of an algorithm that in simple to explain to students, but that quickly grows in complexity and need for resources. Dealing with the potential interactions between particles in a distribution, the N-body problem would ideally compute all possible interactions, and if every N objects interacts with every other (N-1) objects, this results in N(N-1) total possible interactions, growing as N^2 in complexity. This category of problems has applications in many fields such as astrophysics, molecular dynamics, fluid dynamics, and plasma physics.[2] This paper applies the N-Body problem to the simulation of celestial bodies in space, particularly in the case of the "universe in a box" problem where the space being calculated is assumed to be one unit cell out of an infinite expansion.

This research opportunity arose through my selection into the Blue Waters Undergraduate Petascale Education Program. In order to best prepare for the experience we received two weeks of intensive training at the National Center for Supercomputing Applications (NCSA), via the University of Illinois at Champagne Urbana. During the training we were exposed to the world of high performance computing and its architectures and applications. We gained experience in shared-memory parallelism with OpenMP, and distributed system parallelism with MPI. In addition we were exposed to modern applications in computing using GPU architectures. At the time of my research I was a rising junior in Kean University's Center for Science Technology & Mathematics program, majoring in Computational Applied Mathematics. My research and study was done under the mentoring of my advisor, Dr. David Joiner, Kean University. Prior to this summer I had taken multiple mathematics courses, a calculus-based physics course, and a basic java computer programming course.

2. BACKGROUND
2.1 The N-Body Problem
The N-Body problem is a classic physics problem dealing with the interactions of particles. When we are dealing with more two or more particles, each potentially interacts with every other particle, and each force pair is equal and opposite giving N(N-1)/2 unique forces. Unfortunately, the scaling then leans towards N^2 which quickly accumulates when dealing with problems of large N. The initial conditions m_i, r_i, v_i are the mass, coordinates, and

velocity of each particle, respectively. Because of the singularity in the forces of close interactions a cutoff radius, sometimes call a shield radius, is typically used to reduce the computational error due to close interactions. Alternatively, a softening radius can be added in the calculation of the potential. Both of these are options in GalaxSeeHPC [1]

Shield Radius:

$$a_i = \frac{\vec{F}_i}{m_i} = -G \sum_{j=1;j\neq i}^{N} \frac{\alpha_{ij} m_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3}$$
$$\alpha_{ij} = \begin{cases} 1 & |r_i - r_j| > r_{cutoff} \\ 0 & otherwise \end{cases} \quad (2.1)$$

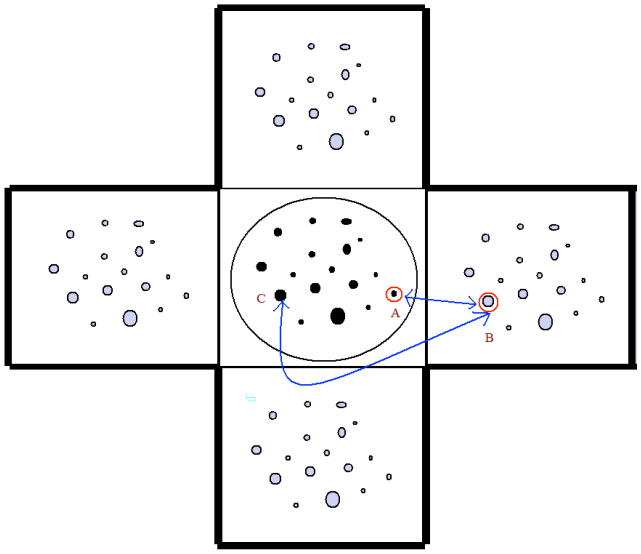
Softened Potential:

$$a_i = \frac{\vec{F}_i}{m_i} = -G \sum_{j=1;j\neq i}^{N} \frac{m_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j + \varepsilon|^3} \quad (2.2)$$

(2.1)

2.2 Particle-Mesh Method

In the case of the "universe-in-a-box" problem, the N-body problem is solved in a system with a "wrapped" geometry, assuming that we are solving for a typical unit cell out of an infinite expanse. The wrapped geometry in gravitational dynamics is typically used to solve for large scale structure, where solving for the entire universe is not practical, and one assumes that there is mass immediately outside of the box that would affect the evolution of the system. Such methods are also used in molecular dynamics calculations, where it is necessary to maintain a consistent bath of solvent molecules around a protein being studied. The Particle-Particle Particle-Mesh method allows for the use of a spectral technique assuming a periodic solution.



(2.3) Illustration of a "wrapped" world as is pertains to the code. The distance between particle A and B represents the "ghost" distance. Particle B does not exist because we are dealing with a "wrapped" world. Therefore, the "real" distance is between particle A and C.

Poisson's Potential Energy equation is given by:

$$\nabla^2 \Phi = 4\pi G \rho \quad (2.4)$$

The algorithm for the density population first begins with incrementing through the array of coordinates and setting the initial density grid to zero at each particle. Next we must find the nearest grid point for each body of mass and "wrap" if needed. After locating the nearest point the range around that grid point is then calculated and we ensure that coordinates are broken by the wrapping of any particles. The final part of the density distribution is incremented through each of the bodies and updating the density array with the values for the particle mesh.

Through a Fourier Transform, the solution to Poisson's equation becomes:

$$\hat{\Phi} = \frac{-G}{\pi |k|^2} \hat{\rho} \quad (2.5)$$

In practice, an additional function is used to smooth out shorter range forces and control the scale of the long range forces. This function, called the influence function, is typically taken to be the Fourier transform of the density function of a single particle in the system during the density grid population [4].

$$\hat{\Phi} = \frac{-\hat{\lambda}(k) G}{\pi |k|^2} \hat{\rho} \quad (2.6)$$

The basic procedure of the PM method is as follows: [6]

- Transform particle mass into density distribution
- Solve Poisson equation using FFT algorithm
- Calculate the force fields and interpolate to find forces
- Integrate to obtain positions and velocities
- Update time step

The Particle-Particle, Particle-Mesh (P3M) is a hybrid method for approximating the solution to the N-body problem. PM methods suffer from an inability to predict short range forces accurately, as any nearby effects are "smoothed" out over nearby gridpoints. The P3M method adds a nearest neighbor "particle-particle" interaction that essentially divides the forces into short-range forces, and long-range forces.[7] Short-range forces are calculated using the direct force method which is the brute force calculation using equation 1.1 from above. The longer range forces are then calculated using the Particle-Mesh approximation.[6] Getting an accurate solution efficiently requires careful selection of the size scale of the density function

for each object in the density population, the influence function, and the radius used to determine nearest neighbors. The hybrid P3M method ideally results in a scaling of $N\log(N)$.[4]

2.3 GalaxSeeHPC

GalaxSeeHPC is a N-Body simulation source code. It requires an UNIX-like environment, or a Cygwin-like environment for windows applications. The code provides many options of different force calculation techniques. You may chose the direct force method, the Fourier Transform based P3M method or the Barnes-Hut tree-based method. For the purpose of this paper we will only discuss in detail the P3M method.

2.4 Message Passing Interface

With the occurrence of larger and larger problems needed to be solved, comes the use and application of supercomputers. The most widespread accepted parallel programming language is the Message Passing Interface, or MPI.[8] Users of C or Fortran can use MPI to pass messages between the nodes of the cluster. This communication can spread out the computer's work and drastically alter the overall performance. If a code has to little work or data the addition of MPI could potentially hurt performance. Also, one must take into consideration the amount of communication the nodes will have to have with each other. Too much communication can also produce harmful effects on a code.

3. PROFILING AND DESIGN

3.1 Profile

In order to attempt to optimize the performance of the code we first began profiling GalaxSeeHPC under many different conditions. The code was examined for various values of N, Final Time, and grid size (the resolution of the mesh, higher the resolution the more accurate). For use of GalaxSeeHPC it is best to have the grid size set to a power of two; for example, 16, 32, 64, etc.

The implementation of the P3M algorithm in GalaxSeeHPC follows three steps in the force calculation, population of a density grid, FFT solution of Poisson's equation on that grid, and interpolation of forces from the Poisson solution back to individual points, along with directly calculated nearest neighbor corrections. Shown below is the percentage of the wall time (as determined through the use of both hard coded timers and gprof) taken up by the density population step. Note that for smaller grid sizes, population of the density grid dominates the time required for a force calculation—compared to the time required for the FFT calculation for larger grid sizes. This suggests that for lower resolution P3M grids, speedups on the order of 20 to 50 times should be obtainable through parallelizing the density population alone.

GRID SIZE	N = 1,000	N = 5,000	N = 10,000
16	.97	.97	.96
32	.93	.94	.94
64	.73	.88	.91
128	.27	.56	.68
256	.08	.18	.24

(3.1) This figure portrays the percentage of total wall time which the creation of the density grid was responsible for.

3.2 Parallelization

By examining the code further, clearly the last part of the density creation loop had the highest potential for parallelization. The statement we parallelized is said to *embarrassingly parallel.* A code is said to be *embarrassingly parallel* if there are no channels between tasks and each process can perform its duties without communication with any other processes.[8] These types of algorithms are usually the easiest to parallelize because of this lack this ability to do work without interaction.

The next step after defining the subset of code to be parallelized is design. A round-robin technique was used, where each node does some work, and returns to do more based on its rank and world size. The rank is each processor's own unique ID number, so that we have a way of distinguishing between nodes. The world size is the total amount of processors initialized at run time. The following three functions are the three most basic and important functions when using the Message Passing interface.

MPI_Init (&argc, &argv);

MPI_Comm_rank (MPI_COMM_WORLD, &rank);

MPI_Comm_size (MPI_COMM_WORLD, &world_size);

Mpi_Init is the function initializes the use of Message Passing Interface. The MPI_Comm_rank and MPI_Comm_size determine the processor id's and total number of proccessors, respectively. Rank and world_size are arbitrary variable names which hold the IDs and total size. Now that we have MPI initialized we can parallelize the code.

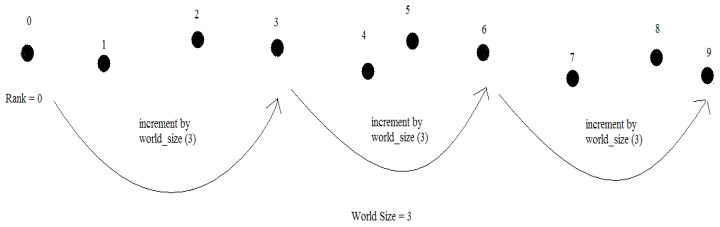
The initial parameters for the loop in the density grid creation was as follows:

For(l = 0; l < theModel->n; l++)

This is a very basic loop incrementing by one and looping through all the bodies of mass in the model. In using the round robin technique we change the loop to the following:

For(l= rank; l<theModel-n; l += size)

Now each processor will begin its loop at its rank (ID) and increment by the world size. The following diagram depicts the round robin use of MPI.



(3.3) Depicts the round robin technique in a group of 3 nodes. Each node begins at its own ID and increments by the total amount of nodes being used.

This type of parallelization is also referred to as data decomposition. This is defined as when multiple tasks have same responsibility but work on different sets of data.[3] As opposed to functional decomposition where each task has its own set of responsibilities.[3] Now the data is broken up, but we still need to make sure every process receives the results of every other process so that have the results of the entire density grid.

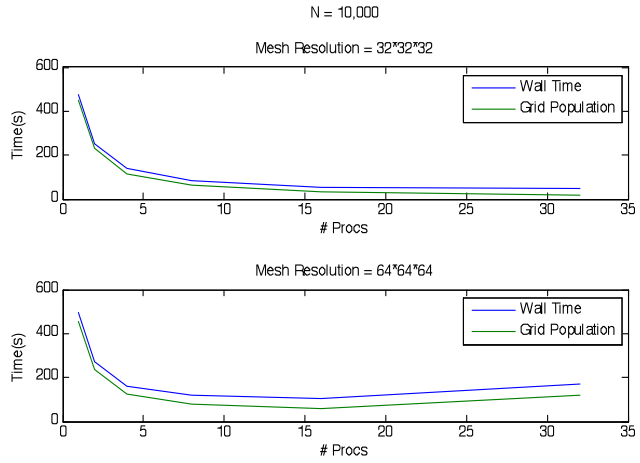
For this we will use one of MPI's reduction commands. MPI_Allreduce is a collective communication, so each processor will receive the reduction results.[7] Below is the command along with its necessary parameters.

```
MPI_Allreduce(
        Void* send_buffer, // = the send buffer
        Void* recv_buffer = the receive buffer
        Int cnt = the number of elements to reduce
        MPI_Datatype dtype = Element type
        MPI_Op op =  Reduction Operator
        MPI_Comm comm. = Communicator
                );
```

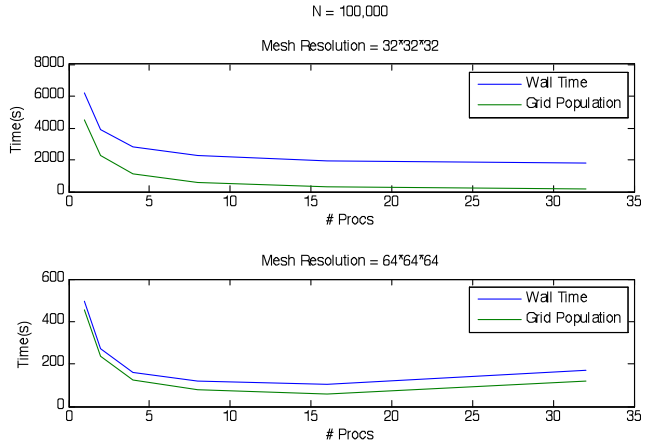
After reducing to a temporary buffer we can transfer all the results back to each proccessors density array. The result is all the nodes sharing the results of the density grid creation with each other.

4. RESULTS

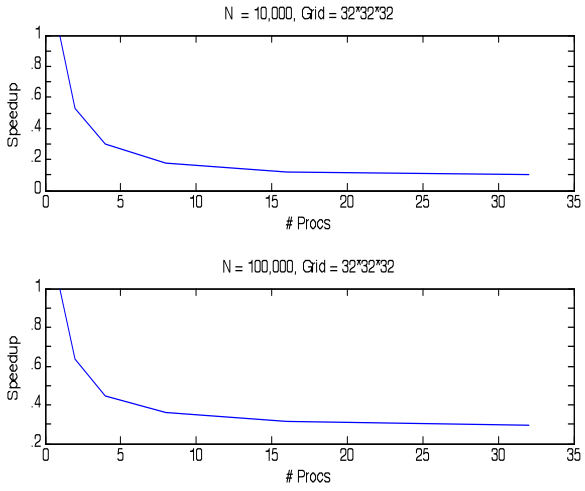
After successfully profiling and parallelizing the code we were able to see very significant results. Below are a few graphs represents the increased speed up we received by running the code with various amounts of nodes.



(4.1) Above are the optimization results when simulating a galaxy of 10,000 stars. Below are the results when simulating 100,000 stars.



In the supercomputing world there exists Amdahl's law which accurately predicts the maximum speedup of parallelized code. Amdahl's law assumes we are trying to solve any problem of constant size as quickly as possible, and can determine potential speedup achievable as you increase the number of processors. [7] The equation is simply that the maximum speedup is equal to one over the fraction of the time spent in code that must be run serially. For instance, if 10% of your compute time is from code which can not be parallelized, then the maximum performance you can achieve is 1/.10 or a speed up of 10x.



(4.2) Above are the Speedups of the Wall-times of GalaxSeeHPC as a function of the # of processors. The speedup value is calculated by divided the wall-time of P processors by the wall time for one processor.

By examining graph (4.2) we see that our parallelized code for 10,000 bodies maxes out at around .1, which results in speedup of 10x. However, when we are dealing with 100,000 bodies we only see a max of .29 with a speedup of 3.45x. We can take our results from the above table (3.1), and apply them to Amdahl's Law. Therefore, for N = 10,000 bodies and a grid size of 32*32*32, the maximum achievable speedup = (1) / (1-.94) or 16x. Although we received a very impressive speedup of 10x for N = 10,000, we weren't able to totally optimize within accordance of Amdahl's Law's projected maximum speedup of 16x.

If we examine table (3.1) further we notice as the grid resolution becomes larger and larger (which increases accuracy), the percentage of total time used in creating the density grid decreases. This is because as the grid sizes increase so does the time in calculating the Fourier Transform by the FFTW algorithm. Therefore, our next task in fully optimizing performance of GalaxSeeHPC should be the profiling and parallelization of the 3 dimensional Fourier Transform of the gravitational potentials.

5. CONCLUSION

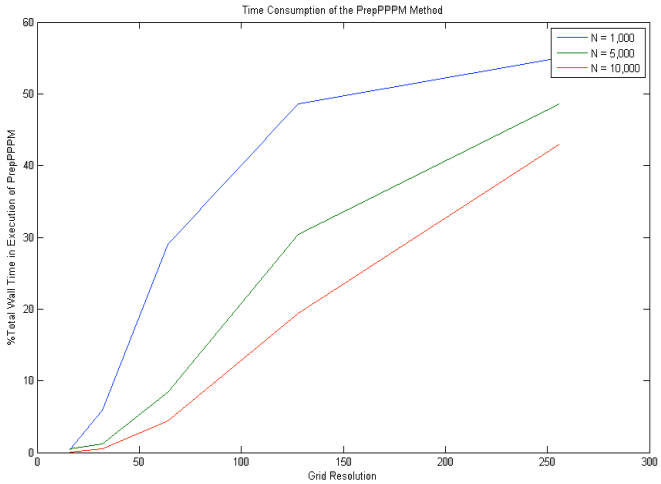
5.1 Analysis

Our analysis shows that the current parallelization of GalaxSeeHPC using the P3M method scales in accordance with expectations, however, limited by the time required to perform the FFT, which as of yet has not been implemented in parallel. The results substantiate that the population of the density grid consumes much CPU time, and that parallelization of the algorithm can lead to improved performance.

Howbeit we achieved speedups of up to 10x; we must not overlook that only one algorithm in the code was parallelized. To obtain maximum performance we must continue to examine the code for potential methods possible to execute in parallel. As stating above, the next step would be optimizing the FFTW algorithm, within GalaxSeeHPC's PrepPPPM method. There is now Message Passing algorithms available for the FFTW open source libraries. Below is data which validates that in order to reach maximum attainable speedups, implantation of the MPI-based Fourier Transforms would be the next sensible phase.

#Procs	1	2	4	8	16	32
%Total Time	3%	6%	11%	17%	22%	25%

(5.1) Above represents the direct proportionality of the percentages of total wall time spent in PrepPPPM, and the number of processors used in execution, for N = 10,000, and a Grid Resolution of 32*32*32.



(5.2) The above graph is the Percentage of total time spent in PrepPPPM as a function of the Grid Resolution. The time in computing the Fourier Transform increases as the mesh size increases.

Use of the GalaxSeeHPC Simulation will require a significant amount of CPU time, and computing power. Time constraints and computing resources necessary will depend on the actual problem set. For instance, computing a simulation of a galaxy of 100,000 stars on one core, over the course of 15 billions years and a time step of .1 million years would require a tremendous amount of CPU time. Our earlier results, when running 100,000 stars for 100 time steps, required just over 6000 seconds to run. Therefore running at 15 billion years with a time step of .1 would require approximately 1500x more CPU time, which results in over 2584 hours!

5.2 Reflections

Through Blue Waters' education program I received an invaluable educational experience. This section describes the impact this research internship had on my learning experience, and how to potentially use this paper in undergraduate education so that others may gain knowledge in high performance computing applications.

Firstly, one must master the basic mathematics and physics concepts behind the simulation. The N-Body is calculated by means of Newtown's basic physics equations depicted in (2-1). This governing equation should not take much mathematics knowledge to comprehend, and I already had prior experience with these physics concepts. The most abstract concept for me, which is most likely to be the one misunderstood by students, is the use of Fourier Transforms. A Fourier Transform in a very basic sense is the transformation of one equation from the time domain to the frequency domain. The result of the transform will depict the frequencies of the original equation. Initially I had no knowledge of Fourier techniques, therefore in order to properly understand the algorithms of GalaxSeeHPC; I had to master the concepts behind Fourier transforms and their applications. Therefore, future students who wish to benefit from this paper must first comprehend the concepts behind the simulation.

I also had to familiarize myself with MPI and ways of thinking *parallel* in order to formulate the best algorithm. The training provided by the NCSA allowed me to efficiently discover algorithms in the code which would benefit by the application of MPI and its parallel routines. However, before I actually implemented any code changes, I had to first properly model and profile the code. I began executing the code for many different sets of initial conditions, and used Microsoft Excel and MATLAB in order to make sense of these results. I also learned how to efficiently use gprof (GNU Profiler) which aided in shedding light on which method calls within the simulation actually were taking up the most time. However, in order to analyze the code as accurate as possible, I also implemented my own timing methods into GalaxSeeHPC. Students must learn how to correctly profile and analyze in order to locate the algorithms which are best-fit to parallelize.

I was also fortunate enough to work in a team environment throughout my research. During my training I had a chance to collaborate with the other gifted students who were chosen as Blue Waters Petascale interns, and continued to use them as resources throughout my research by using OpenStudy, an online study group. I also collaborated with my professors at Kean's NJCSTM and some of our graduate students. There is always more than one way of implementing a problem in parallel and having a team environment is very useful in discovering the optimal solution.

This research experience can potentially be replicated by professors for use in their research, as well as in undergraduate education. The code itself can be used in classrooms, and can execute scalable N-Body simulations for different boundary conditions. Using the parallelized P3M algorithm within GalaxSeeHPC, students and researchers can experience speedups of 10-20x, and will have the ability to execute simulations for very large N. Also students may benefit from the research methods I used and how to successfully analyze algorithms, and implement high performance computing techniques. There are many different paths one could take in furthering this research, depending on their field of study, and intended teaching application. For instance, a computer scientist may consider analyzing the Barnes-Hut method for parallelization, an alternative N-body approximation using tree data structures. However, a physicist may be more interested in using the parallelized P3M methods in order to conduct research on the organization of large scale universes, and how their formation.

6. ACKNOWLEDGMENTS

Thanks to Dr. David Joiner of Kean University, for the constant support and mentorship throughout the process. Also thanks to SHODOR and the NCSA for the support and education experience.

7. REFERENCES

[1] Aarseth, S., *Gravitational N-Body Simulation,* Cambridge Monographs of Mathematical Physics, 2003.

[2] Aluru, S., Prabhu, G.M., and Gustafson, J. *Truly Distribution Independent Algorithms for the N Body Problems*, Ames Laboratory Department of Energy, Ames, Iowa, http://www.scl.ameslab.gov/Publications/Gus/N-Body/N-Body.html.

[3] Binstock, A,. *Data Decomposition: Sharing the Love and the Data*, http://software.intel.com/en-us/articles/data-decomposition-sharing-the-love-and-the-data/, November 4th, 2009

[4] Board, J, and Toukmaji, *A., Ewald Summation Techniques in Perpestive: A Survey,* Computer Physics Communications Volume 95, Issues 2-3, June 1996, Pages 73-92, 1999

[5] Graps, A., *N-Body Particle Simulation Methods,* http://www.amara.com/papers/nbody.html

[6] Shodor, *GalaxSeeHPC*, http://shodor.org/petascale/materials/UPModules/NBody/, 2010

[7] Splinter, R., *A Nested Grid Particle-Mesh Code for High Resolution Simulations of Gravitational Instability in Cosmology,* ftp://asta.pa.uky.edu/cosmology/ngpms.ps

[8] Quinn, M., *Parallel Programming in C with MPI and OpenMP*, 2004

An Automated Approach to Multidimensional Benchmarking on Large-Scale Systems

Undergraduate Petascale Education Program

Samuel Leeman-Munk
Earlham College
801 National Road West
Richmond, Indiana 47374
leemasa@earlham.edu

Aaron Weeden
Earlham College
801 National Road West
Richmond, Indiana 47374
amweeden06@earlham.edu

ABSTRACT

High performance computing raises the bar for benchmarking. Popular benchmarking applications such as Linpack[4] measure raw power of a computer in one dimension, but in the myriad architectures of high performance cluster computing an algorithm may show excellent performance on one cluster while on another cluster of the same benchmark it performs poorly.

For a year a group of student researchers at Earlham College in Richmond, Indiana worked through the Undergraduate Petascale Education Program (UPEP) on an improved, multidimensional benchmarking technique that would more precisely capture the appropriateness of a cluster resource to a given algorithm. We planned to measure cluster effectiveness according to the thirteen dwarfs of computing as published in Berkeley's parallel computing research paper [1]. To accomplish this we created PetaKit, a software stack for building and running programs on cluster computers.

Although not yet the benchmarking suite of thirteen programs that its creators set out to make, PetaKit has become a framework for benchmarking any command-line based program. In PetaKit's construction learned about the challenges of running programs on shared HPC systems, developing techniques to simplify moving software to a new cluster. In addition, we learned important time management skills, specifically by managing our time between classes and our PetaKit work. These skills and accomplishments have been of tremendous benefit to us in our post-baccalaureate careers, and we expect they will continue to be so.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education*; D.2.8 [**Software Engineering**]: Metrics—*performance measures, benchmarking*

General Terms

Parallel, Distributed Memory, Shared Memory, Benchmarks, Human Factors

Keywords

Blue Waters Undergraduate Petascale Internship, Parallel Computing, Parallel Benchmarking, Benchmarking

1. INTRODUCTION

In 2011, the University of Illinois will be home to what will likely be among the fastest publicly-accessible supercomputers in the world. This computer, Blue Waters, will reach sustained operation speeds measured in PetaFLOPS, 10^{15} floating point operations per second. Supercomputers like Bluewaters use a high degree of parallelism to achieve this level of performance, combining the power of huge numbers of computers to do what one computer could not. The catch is that such a high degree of parallelism creates an entirely new set of issues for software designers and requires an approach to programming that for many people is entirely new and foreign.

Despite its tremendous importance, this approach to programming is sorely undertaught in typical undergraduate curricula. Few students are taught to think in parallel, and even fewer are trained in the development of large-scale parallel applications.

The Blue Waters Undergraduate Petascale Education Program, led by Shodor, a non-profit computational science education foundation based in Durham, North Carolina, aims to teach young programmers the skills necessary to harness the power of High Performance Computing (HPC), and therefore Blue Waters and computers like it. For a year, an accepted student works full time during the summer and part time during the school year on a project relevant to HPC.

This paper describes our project and experiences as two of the six interns nationwide to participate in the first year of the program.

2. RELEVANT WORK

Since the late '80s the HPC community has been aware of shortcomings in the popular methods of benchmarking as applied to HPC. The Linpack benchmark, currently used to

rank supercomputers in the Top500 supercomputers list[5], has been repeatedly criticized for imprecisely representing cluster abilities with one flat metric[2][3][8]. Network bandwidth, latency, memory, number of cores on a chip, number of chips on a node, number of nodes on a blade, blades on a rack, all these factors have significant effect on program performance, and certain programs are affected more than others. Linpack, a linear algebra suite, is good for predicting how linear algebra-heavy programs will run on a cluster, but translates more poorly to other programs.

Most notable of the previous work done in multidimensional benchmarking, or judging a cluster with many different statistics rather than just one, is the High Performance Computing Challenge suite (HPCC). HPCC uses seven programs, selected so as to measure important factors of HPC machines such as steepness of memory hierarchy, network latency, and network bandwidth[8]. These make for detailed analyses for directly comparing cluster computers. A computational scientist with relatively little HPC experience looking at all these factors, however, would likely have trouble translating such statisics into an estimate of the performance of his or her own program. Application benchmarks, such as those suggested by the numerical aerodynamic simulation benchmarks project(NAS) and the performance evaluation for cost-effective transformations activity (the Perfect benchmarks), would use programs closer to real science applications to benchmark the cluster, generating benchmark data more relevant to real research[2][3].

Although common scientific programs have clear advantages as meaningful benchmarks, they suffer from issues of portability. Scientific software used in professional research tends to be large and demands careful tailoring to a specific system for maximum optimization. Devoting a large research task to porting scientific software generally costs more time and money than the resulting benchmark is worth [2]. NAS specifically mentions the absence of automatic tools to ease porting scientific applications. Today such tools, although far from trivializing the porting process, do make the prospect of application benchmarking much more affordable.

3. RESEARCH PLAN

Our goal was to build a framework that made application benchmarking feasible for evaluation of clusters' appropriateness for particular sets of scientific software. Admittedly, automated portability for arbitrary programs is not yet a reality, but we could make the benchmarking process much easier nonetheless. With our system we would be able to easily make a set of applications into a set of application benchmarks.What set of applications to use was not so simple. For the purpose of predicting performance on clusters, programs can be judged by their patterns of communication and ratios of communication to computation. A group of Berkeley researchers analyzed communication patterns among parallel programs and wrote a paper describing thirteen "dwarfs," or computing paradigms. According to Berkeley, any conceivable program may be described as a combination of one or more of these dwarfs[1]. Our system would take representatives of each of these thirteen dwarfs and collect performance data on various clusters. Then a user would identify his or her program using these paradigms and have the opportunity to make an educated decision of what cluster to use for his or her research.

Our system, PetaKit, would handle the entire process, from deployment and compilation on the remote host to visualization of the performance statistics in web-browser format. After decompressing the PetaKit file on the cluster of choice, the user proceeds into the generated folder and runs ./config to build all the necessary files with GNU Autotools, a system for managing compilation on various different systems. Then the user runs the performance data harvester perl script, stat.pl, which takes various parameters including style of parallelization, problem size[1], number of threads or processes to spawn and processors on which to spawn them. Stat.pl then runs these programs and sends various information, including number of threads, problem size, and wall time to a PostgreSQL database on a computer at Earlham College. A browser-based visualization tool converts user selections of the data into a graph.

To build the framework, we started with representatives of two of the thirteen dwarfs[1]. These are area-under-curve (AUC) – a definite integral approximation program based on the Reimann sum and the MapReduce dwarf, and GalaxSee – a galaxy simulation that serves as an n-body problem. AUC gives each thread (in some cases each thread is in its own process) a fraction of the area under the curve to estimate, they work and each sends its answer to the main thread, which sums the answers up and prints the total area under the curve. Computation increases linearly over problem size while communication remains constant. GalaxSee, on the other hand, splits the stars among the threads and each calculates the new positions based on the positions of all the other stars. Then the new positions are collected and redistributed and the next set of positions is calculated. Communication and computation both increase at a rate of $O(n^2)$ where n is the problem size. Each of these programs was split into four "styles" of parallelism: serial, shared memory (OpenMP), distributed memory (Message Passing Interface, MPI), and hybrid shared and distributed memory. The idea of having all four styles instead of just the most effective one was to confirm that hybrid was the most effective, and could outperform MPI under the right circumstances. In addition to resolving that issue, our expectation was that at the end of this project PetaKit's graphs would teach us something about the difference in scaling between MapReduce and n-body problems on various clusters.

4. CHALLENGES AND OPPORTUNITIES

Of all the challenges our group faced, most pervasive was the difference in the cluster environments that we came across when attempting to build and run our code. Each of about five clusters proved to be unique in its combination of compiler, network interconnect, linker, file system, scheduler, and software stack. Each error that occurred during building or running of our code had to be pinpointed as a problem with our code (either in its functionality or our fundamental

[1]Problem size is the variable in a program that has the most significant impact on the program's runtime. A Reimann sum algorithm, for example, uses the number of segments into which it splits the area under the curve as its problem size, and a galaxy simulation uses the number of stars in the simulation.

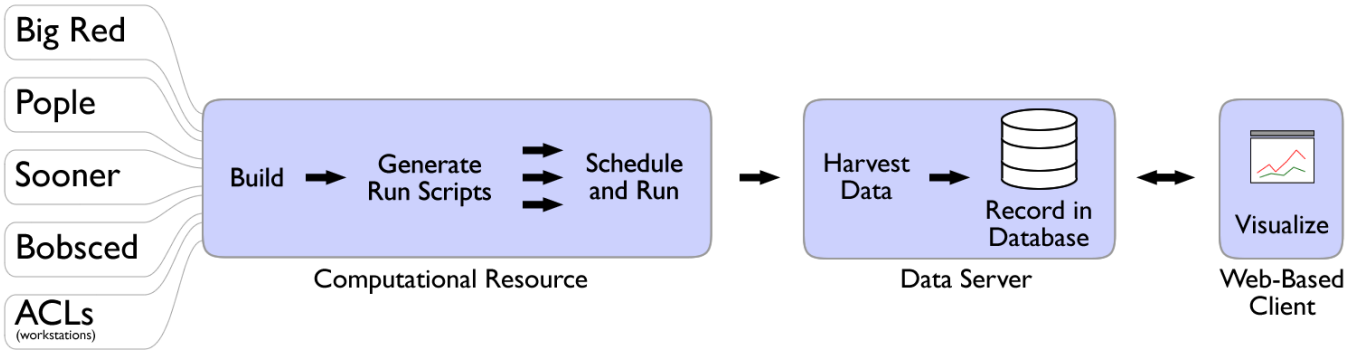


Figure 1: PetaKit's flow of data[6].

understanding of it), the compiler options we were attempting to use, a typo in the configure script, or a myriad of other possibilities. New, unrecognized errors forced us to perform a great deal of trial and error until we were able to find clues as to the sources of the errors. As we encountered more errors, however, we became more adept at recognizing common points of failure and remedying them. We were also able to make our package more robust by guarding against possible error scenarios. In this way, then, we learned a great lesson in working in new cluster environments, that there are many errors that can arise on the new system, and one must use the skills one has learned in the past but also have the courage to attempt new solutions.

A particular challenge in working on a shared cluster as opposed to a personal device is the scheduler. On a high-traffic cluster machine an automated scheduler, a program that distributes many tasks among many more processors, is the only way to share resources effectively. Because PetaKit is a benchmarking suite with an emphasis on the effect of increasing parallelism, it has to run the same program many dozens of times to collect all its data. In order to minimize the time each separate run of this program would have to wait before executing, PetaKit submits each run as its own job. In this manner, rather than demanding that the scheduler find one contiguous span of time for all the runs to complete, many small runs submitted to a scheduler could be run one by one on whatever processors become available. Incidentally this also takes advantage of the inherent parallelizability of benchmarking. Each individual instance of a program has no dependencies on or need to communicate with other instances, making it relatively easy to run them in any order or at the same time. The only caveat of running them in parallel is that the visualization program cannot rely on data coming in in any particular order.

Of course, like everything else, schedulers vary significantly by cluster. Designing PetaKit to work successfully with these various schedulers sometimes proved to be more than trivial syntax changes. For instance, The University of Oklahoma's supercomputer "Sooner" runs a Load Sharing Facility (LSF) scheduler, which, unlike PetaKit's other supported schedulers, Portable Batch System (PBS) and LoadLeveler, prints helpful information to standard output as it runs. The issue was that stat.pl collected its data from standard output, and LSF's messages to standard output were confusing it. We eventually bordered the meaningful output with a distinctive string on either side to distinguish it from the rest of the output.

In addition to variations in schedulers, we also had to work with a completely different staff team for every remote cluster we used. Nearly every time we tried to port our code to a new system we would have to find out little details such as the location or version of MPI libraries. First we would look to the cluster's relevant website for the information. When that didn't work, we'd get in touch with the cluster's system administrators or help staff. Generally, we received prompt, helpful replies to our questions, which greatly improved our productivity.

Besides technical challenges, an issue we faced on a deeper level was finding a way to balance the work of the project with our other course work. This was particularly a challenge because of the looser deadline structure associated with the project. Whereas classes were meeting two or three times weekly, our group's meeting to discuss the project only occurred once per week, and even then there were seldom hard deliverables. We had to develop a new level of focus and motivation for the project to be able to stay on top of it. Thus, we learned new ways of motivating ourselves, for example splitting the project into small pieces, working on small goals throughout the week rather than letting the greater end become daunting. Through constructing our own schedules and picking our own goals we were able to find personal meaning in the tasks we completed, and we made more efficient use of the time we could devote to the project.

Throughout the program our mentor Dr. Charlie Peck, associate professor of Computer Science at Earlham, was helpful and available to answer questions. During our weekly meetings we discussed our progress and where we'd be heading next. Dr. Peck would look at our data and point out places it did not make sense, such as when he helped us notice an issue in a graph that eventually led us to a set of old data that was being erroneously combined with recent data and leading to bad graphs.

5. RESULTS

At the writing of this paper, the PetaKit framework exists in its entirety. The system is not yet public, but the steps for a developer to benchmark a supported program on a given cluster are as follows[2] (see Figure 1):

1. Build (Computational Resource)
   (a) Generate a compressed folder from the latest source and send it to a supported cluster
   (b) Decompress the folder
   (c) run `./configure --with-mpi --with-omp` (Autotools) in the folder
2. Run StatKit (Computational Resource)
   (a) Run StatKit, specifying relevant options, such as an identifying tag (see Figure 5)
   (b) StatKit takes the specified parameters over which to observe scaling and iterates over them, building a script file for each combination of parameters
   (c) As it is built, each script is submitted to the scheduler once for each repetiton requested
3. Scripts run (Computational Resource)
   (a) Run program, collect output
   (b) Pipe output through secure shell to centralized data server
4. Data Collection (Data Server)
   (a) Parse meaningful performance data from raw output
   (b) Access PostgreSQL database and insert performance data
5. Data Display (Web-Based Client)
   (a) User selects data by tag
   (b) User selects the independent and dependent data to observe
   (c) User selects a variable over which to "split" the data into multiple lines
   (d) PHP backend accesses database, averages repeated runs, and creates a graph
   (e) Page displays graph image file

PetaKit developers can generate a compressed folder from the latest source and send it to a supported cluster, where Autotools is run to build the files. Then StatKit takes its parameters and generates a set of shell scripts, one for each combination. StatKit submits these scripts to the scheduler, resubmitting for each repetition, and when each is finished it pipes its output through ssh to the parser.pl script on a computer at Earlham College. Parser.pl parses the output into data which it sends to a PostgreSQL relational database. The data remains in the database where it can be accessed via our web browser application which allows the user to pick dependent and independent variables and compare the scaling of one setup to another(see Figure 1). The most common use of the graph is to compare OpenMP performance to MPI performance to hybrid performance by plotting the three threads vs. walltime.

This system has collected and visualized data on over five clusters:

- Indiana University's Big Red, a BladeCenter JS21 Teragrid resource with PPC 970 processors and a LoadLeveler scheduler
- Pittsburgh Supercomputing Center's Pople, an SGI Altix 4700 Teragrid resource with Itanium 2 Montvale 9130M processors accessed via a PBS scheduler
- Univerity of Oklahoma's Sooner, a PowerEdge 1950 Teragrid resource with Xeon E54xx processors and an LSF scheduler
- Earlham College's Bobsced, a decomissioned cluster with Intel Core2 Quad processors and a PBS scheduler
- Earlham's Advanced Computing Lab Computers (ACLs), a group of Dell commodity workstations with Pentium D processors and no scheduler

These visualizations have been presented at TeraGrid and SuperComputing in 2009, and at Earlham College in its 2009 research conference[7]. The most dramatic discovery of our benchmarking software was that the particular programs we were using to test it had serious bugs and fell short of the scaling one would expect of n-body and MapReduce problems, even at large problem sizes. AUC behaved as expected except for its hybrid version. Hybrid's failure to outperform MPI on BigRed was understandable for AUC, a Map-Reduce dwarf with low communication overhead. On Sooner, however, Hybrid AUC took over eight times as long to run as its MPI counterpart even with heavy parallelization and large problem size(see Figure 2). Even accounting for the additional overhead of a hybrid program, such a performance penalty suggests an error. As for GalaxSee, we were still having significant trouble running it and had not yet gotten results. Fortunately, we realized that the tools we had developed to get our results were actually much more important than the results themselves. A poster describing PetaKit as a performance data collecting framework was presented at SIGCSE and the Earlham College Annual Research Conference in 2010[6].

In the summer UPEP workshop at the National Center for Supercomputing Application (NCSA) we presented PetaKit to the new group of UPEP interns. We explained it to the students and guided them through a lab where they equipped their programs to output PetaKit-compatible statistics (using supplied header files) and collected data on it. We had modularized the harvester such that it could output data into a text file to be graphed via gnuplot, but StatKit was still a long way from being truly user friendly. Some students accomplished this with relative convenience, but many

[2]This is one possible flow of data chosen to show every piece of the project. By default the harvester saves its data to a text file that can be directly visualized with the accompanying plotting program PlotKit.

had trouble using a PetaKit interface that was not yet designed for benchmarking arbitrary programs. By the next presentation at the University of Oklahoma (OU) the harvester gained improved versatility via a "template system" where the command line into which parameters are dropped was no longer hardcoded but defined by the user at runtime with the command `--cl`. Since then, scheduler submission scripts have also been templatized, allowing users to make a "script template" for a given system and instruct StatKit to use it with `-t`. Users can send their template scripts to us, and we will incorporate them, expanding PetaKit's cluster compatibility. For further user-friendliness we added a feature to PetaKit,`--proxy-output`, to collect minimal data from programs that have not had PetaKit output added to their code. To minimize visualization hassle we now include a supplementary program, PlotKit, another Perl script that automatically averages repeated runs, organizes the data, and creates and runs a gnuplot script to make a visualization according to a few simple parameters the user gives.

Using this improved PetaKit system it became simple for us to try many different programs without wasting our time reprogramming either them or PetaKit. With our new system we quickly tried a few different versions of GalaxSee until we found one that effectively used MPI to perform faster(see Figure 4). Now we can compare the results of AUC and GalaxSee on Sooner. While AUC shows a relatively smooth asymptotic curve, GalaxSee's heavy communication load makes it more sensitive to architecture changes. We see this in the graphs' bumps, the most prevalent being in the jump from eight to nine cores where GalaxSee added an extra node and started sending messages over a network.

6. IMPACTS

With the information we learned from our project, we will continue to improve PetaKit as both a software scaling evaluation tool and a cluster evaluation tool. In terms of the cluster evaluation our first step will be to debug GalaxSee and hybrid AUC. Then we'll add the other eleven dwarfs, making sure to check the efficacy of each before committing it. The UPEP workshop gave us the opportunity to show other people PetaKit, people who will likely be programming parallel algorithms that could very well find their way into PetaKit itself. Collecting pre-made programs will be a much more effective way of building PetaKit's dwarf array than building each one from scratch, so these contacts will be extremely useful.

Once we've shown the efficacy of hybrid, continuing to make four versions of each program is unweildy and unnecessary. Instead we will design a program to make the best use of the resources available, for example a hybrid MPI-OpenMP program that, when run on a dual-core laptop computer behaves as plain OpenMP. At the UPEP workshop we became acutely aware of the growing popularity of the Graphics Processing Unit for General Purpose Computation (GPGPU), so another version of each program may be introduced where it is appropriate to support GPGPU, or other accelerators as they become common on cluster systems.

Although Dr. Peck always made a point of addressing parallel processing in his computer science classes, we went into our UPEP internship with only the most basic understanding. Some of us were under the impression that we knew everything there was to know about the Unix command line, but writing Perl and shell scripts solving the complex problems PetaKit presented we learned to harness Unix's true capabilities, far beyond what in simple academics we had taken for granted.

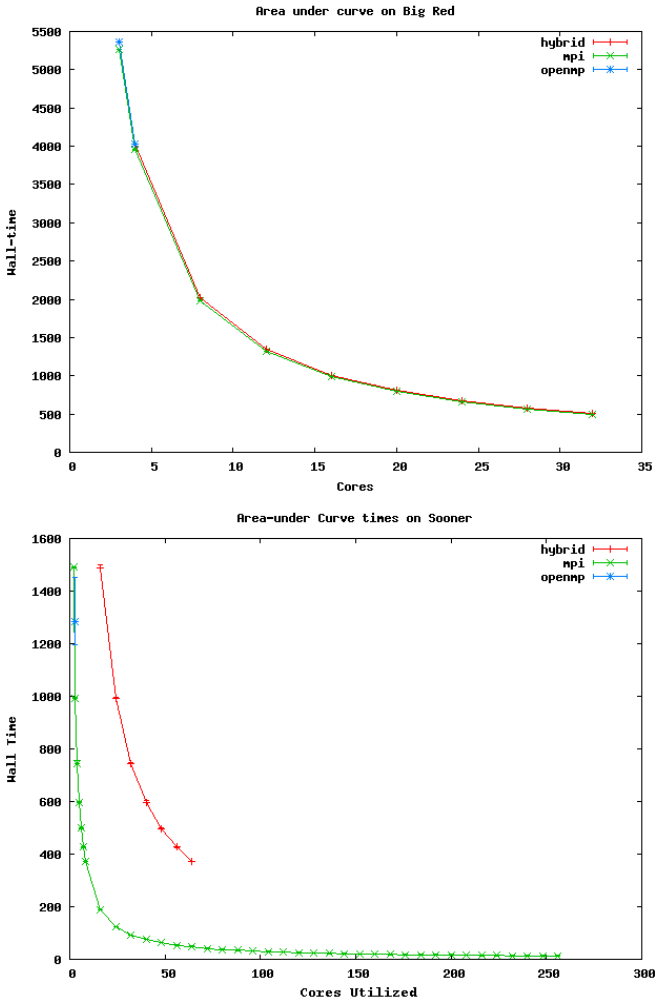


Figure 2: PetaKit visualization of AUC scaling on Sooner and Big Red[6].

The skills we learned in UPEP translated easily to much of our classwork. After building the statistics harvester for PetaKit, the test automation portion of software engineering class came as second nature to Sam. He ended up spending most of his time helping his classmates with their automation projects. The time management skills learned during the internship helped Aaron in his senior seminar and independent study. Deadlines were much more spread out than in normal classes, making a structured working schedule essential for success. Where other students struggled to schedule their time such that they were consistently doing work, Aaron felt comfortable staying focused and motivated under the looser schedule of the seminar. Both of us learned the power of source code control, something that seldom comes up in undergraduate computer science classes. Aaron made

```
 perl stat.pl
-t al-salam.sh --scheduler pbs
--cl 'mpirun -np $processes
 ~/area-under-curve/area-mpi -s $problem_size '
--processes 2-4-16 --problem_size
 1000000000,100000000000 --proxy-output
```

Figure 3: An example StatKit command line

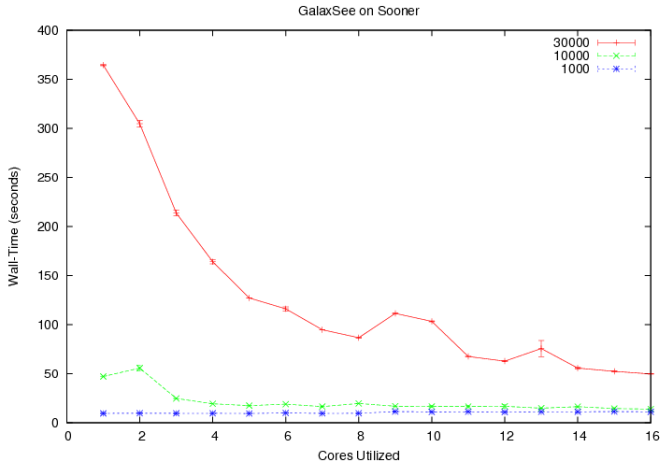


Figure 4: StatKit and PlotKit visualization of various problem sizes of an MPI GalaxSee program.

heavy use of Github in his senior year independent study and Sam was quick to suggest collaboration via Github to his group in robotics class.

Of course, besides everything else we learned, UPEP's goal was to teach us parallel thinking, and we have. Now the second question after "How would we solve this problem algorithmically" is always "How would we do it in parallel?" Far from being daunting and byzantine, parallel programming is just another tool in our repertoire. We're now interested in helping other people who find parallel programming intimidating to learn how much they really can do with it if they just give it a chance.

7. RECOMMENDATIONS

The UPEP program was a powerful experience. For many of us, it was unlike any before. Between talented mentors, open-ended project assignments, and access to a vast array of HPC professionals and educators, scholarly HPC literature, and of course full-scale shared cluster systems that we shared with genuine high performance scientific models, it taught us much more than a classroom environment could.

Certainly, UPEP's success, at least for us, was in significant portion due to Dr. Peck, our motivated, high-energy mentor who was not only happy to lecture us and other members of our small Earlham College Cluster Computing group in our weekly meetings, but gladly volunteered his time to give each and every one the individual help he or she needed. Even as we attempted to solve problems which no one had before attempted to tackle, Dr. Peck guided us with a calm, confident demeanor, not allowing a shred of doubt that success was assured.

For the most part, Dr. Peck took a non-interfering role in mentoring our internships. He would send us off with general tasks at first, then later in the internship he stepped back, remaining available for help, asking during meetings only whether we had enough work to keep us going. Given the opportunity to explore the project on our own terms, we developed a sense both for how to structure our work and what areas within HPC interested us most, whether they be analyzing and rewriting parallel code or constructing an effective, user-friendly interface for generating and sending out large sets of performance tests. As a result, we stayed motivated, working on what we found engaging at our own pace: challenging, but not exhausting. Our self-defined relationship with the project allowed us to connect with it more closely and to feel more accomplished with the results. Rather than some job we were doing for Dr. Peck, the projects became our own. Encouraging future UPEP interns to similarly take the lead in planning their own projects should help them to establish a more meaningful and personalized relationship with HPC.

As we carried out our personalized projects, we did so on actual high-end clusters that provided tools, libraries, and challenges for developing and running parallel code. Such access allowed us to put theory into practice and to experience the development of real parallel software on actual computers. Continuing to provide access to high end clusters through resources such as the TeraGrid and Blue Waters will help interns gain real hands-on experience with developing high end parallel code. Though much of working with clusters and parallel code appears daunting at first, the inherent challenges, more than simply justified by the corresponding educational benefit, are an integral part of the education themselves. To paraphrase HPC educator and system administrator Henry Neeman, "People should learn in education to do what they will do in real life." [1]

Continuing the theme of real-life connection, Dr. Peck provided us with scholarly work in the field, and encouraged us to communicate with experts who deal with high performance computing in their work, whether in industry or academica. Although practical experience working with clusters and parallel code clearly belongs at the center of UPEP, the encounters we had with professional-level HPC people and materials – the Berkely view paper [1], our work with Dr. Peck, and the people we met at conferences – served as important supplements to our education, teaching us where our work fits in the greater practice of HPC. Armed with our understanding of our context, we could improve the quality and depth of our research, for instance using Berkeley's thirteen dwarfs as guides for our benchmarks. Future interns will benefit from greater networking and connection to HPC researchers and workers.

Punctuating our internship with the occasional conference served a dual purpose: not only to allow us to see the work of others, but to present our own work as well. Each of the presentations we made at conferences and workshops helped us to share our project and receive meaningful feedback from experts in the field. Furthermore, the preparation for these

presentations forced us to reflect on our project and shape an informative and concise defense of it. These presentations were so effective at getting us to evaluate our standing that, as mentioned before, at one point in presentation preparation we actually changed our direction, going from a display of the data collected from poorly constructed programs to focusing heavily on the sophisticated system that we were developing for collecting that data. Regular presentations will encourage interns to feel comfortable synthesizing, explaining, and defending their projects, which will lead to improved comprehension and learning.

For all its strengths, the UPEP program's first year exhibited some disorganization characteristic of a nascent program with no preexisting documentation or buearacracy. None of the issues were particularly troublesome, but we were glad to find them addressed in the first official UPEP workshop at the National Center for Supercomputing Applications at the end of our internships (and the beginning of those of the next year's interns).

The UPEP workshop was not only the first time we met the next generation of UPEP interns, but the first time we met the rest of our generation of UPEP interns. Had we been in touch with each other from the beginning we could have communicated and helped each other with our problems in our projects.

Fortunately, the workshop resolved many of the human networking issues. The workshop was extremely informative, and working so close to Blue Waters itself made for an ideal beginning of the UPEP internships. Meeting so many people with the same interests, especially well-spoken presenters, was inspiring and motivating. In general, having the previous year's interns help teach the next year's is an excellent strategy. It's a cost-effective solution that solves the networking issue while strengthening the expertise of both parties.

8. CONCLUSIONS

Although the concept of a universal application benchmark made up of an instance each of Berkeley's thirteen dwarfs is still a long way from realization, the work done on the PetaKit project has by no means gone to waste. PetaKit's daughter project StatKit has come into its own and even generated its own companion project PlotKit. StatKit and PlotKit are currently being used in UPEP parallel education curricula.

Our work in UPEP was extremely rewarding, as was developing and presenting curricula for the UPEP workshop at the end of our internship. Sam has accepted a job at Shodor, where he continues work on PetaKit and other STEM and HPC education projects. He is looking into Carnegie Mellon and University of Southern California to get his PhD in Natural Language Processing, an often computationally intense field that will benefit from effective parallel experience. Aaron hopes to pursue a PhD in computer science and to have the opportunity to use the knowledge gained at UPEP to teach and mentor others. He is currently working as a system administrator in Earlham College's Computer Science department, putting to use the skills he has obtained from working on a variety of cluster implementations.

9. REFERENCES

[1] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks. *SC Conference*, 0:158–165, 1991.

[3] G. Cybenko, L. Kipp, L. Pointer, and D. Kuck. Supercomputer performance evaluation and the perfect benchmarks. *SIGARCH Comput. Archit. News*, 18:254–266, June 1990.

[4] J. Dongarra, P. Luszczek, and A. Petitet. The linpack benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, 15(9):803–820, 2003.

[5] J. J. Dongarra, H. W. Meuer, E. Strohmaier, J. J. Dongarra, H. W. Meuer, and E. Strohmaier. Top500 supercomputer sites. Technical report, Supercomputer, 1997.

[6] M. Edlefsen, A. F. Gibbon, B. Johnson-Stahlhut, D. Joiner, S. Leeman-Munk, G. Schuerger, A. Weeden, and C. Peck. Collecting performance statistics on various cluster implementations. Poster at ACM Special Interest Group on Computer Science Education, April 2010.

[7] M. Edlefsen, A. F. Gibbon, B. Johnson-Stahlhut, D. Joiner, S. Leeman-Munk, A. Weeden, and C. Peck. Parallel performance over different paradigms. Poster at Teragrid Conference and Supercomputing Conference, 2009.

[8] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi. Introduction to the hpc challenge benchmark suite. 2005.

Notes

[1]H. Neeman (personal communication, August 8, 2010).

December 2010

Volume 1 Issue 1

TABLE OF CONTENTS

ISSN 2153-4136 (online)