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].

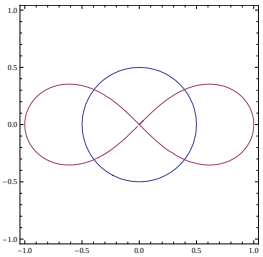

Figure 1: The Four Points of Intersection

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)