# Binomial Solutions to Smale's 17th Problem and their Application to Chemical Reaction Networks

Caleb Xavier Bugg, Morehouse College

July 24, 2014

**Abstract**

In 1998, Stephen Smale proposed a list of eighteen questions for the mathematical community. Smale's 17th problem is concerned with the development of a deterministic algorithm that can approximate a root of a random polynomial system in polynomial-time. This project in its current form provides a positive answer to Smale's 17th problem for binomial systems, and explores a concrete application of the algorithm in chemistry. The main result of this project is an algorithm that approximates a root of an entire binomial system ($n$ variables, $n$ equations) in polynomial-time. The algorithm utilizes matrix exponentiation and the Smith Normal Form of an integer matrix in order convert the system to a simpler system.

Binomial systems are used in a variety of mathematical modeling situations. In particular, there are sufficient, easily verifiable conditions for the expression of a chemical reaction network (CRN) as a binomial system. We utilize our algorithm to solve these systems to determine the steady-state concentrations of the species in chemical reaction networks.

# Contents

**1 Introduction** **3**
1.1 Results . . . . . 3

**2 Background** **4**

**3 Methods** **5**
3.1 Monomial Change of Variable . . . . . 5
3.2 Real Roots and the Bisection Method . . . . . 7
3.3 Matrix Exponentiation and Smith Normal Form . . . . . 9
3.4 Chemical Reaction Networks . . . . . 12

**4 Future Work** **13**

**5 Algorithm** **14**

**6 References and Acknowledgments** **17**

# 1 Introduction

This project in its current form provides a simple algorithm that can be used to approximate the roots of binomial systems. This is the main result of the project as of now, but continuing efforts of the research team can also yield interesting results for systems and computational biologists. Smale's 17th problem calls for the deterministic algorithm to run in time polynomial, at worst. This means that as the input grows (degree and number of variables), the computational complexity grows at time no worse than polynomial to the input (n = number of variables, d = degree). We show that by making bounding every component of the algorithm to run polynomial time (at worst), then the average complexity of the algorithm is polynomial with respect to the input. The methods of this project will be later shown, and background to the problem discussed. This research was conducted in an eight-week time period, resulting in a rather small number of finite application results, despite the wishes and best efforts of the research team.

The results that follow were implemented into an algorithm that was run in Sage, a program that utilizes the Python programming language and contains several useful built-in functions. The software is open-source download, and subsequently the results can be easily reproduced for those interested.

## 1.1 Results

The main result of this project was an easily-readable algorithm that could approximate the root of real binomial systems. It is important to note that the solutions were restricted to the reals because of the application associated with this project. As we will later discuss, $\kappa$ values that denote proportionality constants are strictly positive, resulting in Ordinary Differential Equations (ODE's) with real solutions. The algorithm developed for this project will be written verbatim in section 6, but can be written in pseudo-code as follows:

**Algorithm**

Def binomial system

1. Before initiating algorithm, convert binomial system into form $x^A = c$

Input: $A = [a_{ij}] \in \mathbb{Z}^{n \times n}$ $c = (c_1, c_2, \ldots, c_n)$
Conditions: $x = [x_1, x_2, \ldots, x_n]$ $y = [y_1, y_2, \ldots, y_n]$ $c = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n_+$
Output: Approximate Root

2. Calculate the Smith Normal Form of Matrix $A$. Store $U = [u_{ij}]$ $V = [v_{ij}]$ and $S = [s_{ij}]$

3. Perform a monomial change of variable. Let $y^U = x$, then $y^{UA} = c$ and $y^S = c^V$

4. Determine $y^S$ and $c^V$ with matrix exponentiation program. Hence, $y_1^{\alpha_1} = \beta_1 \ldots y_n^{\alpha_n} = \beta_n$ where $(\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ and $(\beta_1, \ldots, \beta_n) \in \mathbb{R}^+$

5. Solve each equation with bisection method algorithm. Return vector $\tilde{y} \in \mathbb{Z}^n$

6. Calculate $\tilde{y}^U$ in order to return variable after monomial change, producing approximate root vector $\tilde{x}$

7. Test roots in $\tilde{x}$ versus approximate root theorem.

8. End

This algorithm utilizes notation for matrix exponentiation in the preamble (Conditions before initiating the algorithm) which will be thoroughly discussed. Also, the Smith Normal Form of an integer matrix was shown to be algorithmically computable in time polynomial 35 years ago, by Kannan and Bachem [1]. Therefore, the 21st century version of Sage can certainly compute the SNF in time better than polynomial. Lastly, the $y^U = x$ and $\tilde{y}^U = \tilde{x}$ concept is commonly referred to as a *monomial change of variables*, which is simple to show through a later example.

# 2 Background

In the year 1900, German mathematician David Hilbert proposed a list of twenty-three mathematical problems to be solved by the mathematical community in the 20th century. In this spirit, world-renowned mathematician and Fields Medalist Stephen Smale proposed a list of eighteen nontrivial questions for the mathematical community in 1998, with the support of the International Mathematical Union. As to demonstrate the importance of this problem set, the solution to the second problem was deemed worthy of the Fields Medal. This project in its current form provides a positive answer to Smale's 17th question for binomial systems, and explores a concrete application of the algorithm in chemistry. The 17th question is concerned with the development of a deterministic algorithm that can find a root of a random polynomial system in polynomial-time, at worst. This means that as the input grows, the output runs in time polynomial to the input. The main result of this project is an algorithm that could find a root of an entire binomial system ($n$ variables, $n$ unknowns) in polynomial-time. The algorithm utilizes matrix exponentiation and the Smith Normal Form of an integer matrix in order to assign values to variables and trace those values back to approximate roots numerically. Then, Millán et al. (2011) provide sufficient conditions for the expression of a chemical reaction network (CRN) as a binomial system. Hence, the developed algorithm can also be used to determine the steady states of CRN's that meet sufficient mathematical conditions.

We would not like to convey the message that our research efforts towards this problem are the first of its kind, so let us highlight some previous work. In 2008, Beltrán & Pardo provided the first holistic partial positive solution to Smale's 17th problem. They demonstrated a uniform probabilistic algorithm that runs in time polynomial. The algorithm is deemed a "Las Vegas Algorithm" because it will tell the user if it fails to run, rather than producing a false answer. Hence, the algorithm is probabilistic and not deterministic (deterministic algorithms run with probability = 1). In 2012, Cucker & Bürgisser provided a deterministic algorithm for Smale's 17th problem, but it runs in time $N^{O(\log\log N)}$. This complexity is exponential with respect to input $N$, and therefore does not meet the complexity bound to be a complete positive solution to Smale's 17th.

# 3 Methods

The mathematical computations and considerations that correspond to this project's algorithm will now be discussed. Several concepts of programming, linear algebra, algorithmic algebraic geometry, and other disciplines were taught to the undergraduate researchers before starting this program. The process of learning higher-level mathematics, and then applying it, is no easy task. So, the following presented concepts have been discussed tirelessly and many theoretical components (probability and graph theory, to name two) are simplified for this paper.

## 3.1 Monomial Change of Variable

This subsection will begin with the presentation of an example of a monomial change of variables and then highlight the importance of this concept to this project. So given the system:

$$f(x_1, x_2) := x_1^5 - \frac{49}{95}x_1^3x_2 + x_2^6$$

$$g(x_1, x_2) := x_2^5 - \frac{49}{95}x_2^3x_1 + x_1^6$$

We can rescale the variables and divide by suitable monomial terms to get:

$$r(y_1, y_2) := y_1 - 1 + y_2$$

$$s(y_1, y_2) := 1 + Ay_1^a y_2^b + By_1^c y_2^d$$

For some real $A, B$ and some rational $a, b, c, d$,

First, manipulate $f$ such that it is in a suitable form to undergo a variable change to $r$, Then,

$$f(x_1, x_2) := x_1^5 - \frac{49}{95}x_1^3x_2 + x_2^6 \Rightarrow \frac{95}{49} \times \frac{f(x_1, x_2)}{x_1^3x_2} = \frac{95}{49}x_1^2x_2^{-1} - 1 + \frac{95}{49}x_2^5x_1^{-3}$$

At this point we substitute, so: $y_1 = \frac{95}{49}x_1^2x_2^{-1}$ and $y_2 = \frac{95}{49}x_2^5x_1^{-3}$, so now $f$ looks like

$$r(y_1, y_2) := y_1 - 1 + y_2$$

Then, with an established $y_1$ and $y_2$, we manipulate $g$ to look like $s$. So,

$$g(x_1, x_2) := x_2^5 - \frac{49}{95}x_2^3x_1 + x_1^6 \Rightarrow \frac{-95}{49} \times \frac{g(x_1, x_2)}{x_1x_2^3} = 1 + \frac{-95}{49}x_2^2x_1^{-1} + \frac{-95}{49}x_1^5x_2^{-3}$$

But, we have already established a variable change, so we must write this formula in terms of our known $y_1$ and $y_2$. This requires us to equate an $Ay_1^ay_2^b$ to $\frac{-95}{49}x_2^2x_1^{-1}$ and $By_1^cy_2^d$ to $\frac{-95}{49}x_1^5x_2^{-3}$ which is certainly an easy task, which is why formatting hold such importance. Hence,

$$A(\frac{95}{49}x_1^2x_2^{-1})^a(\frac{95}{49}x_2^5x_1^{-3})^b = \frac{-95}{49}x_2^2x_1^{-1}$$

and

$$B(\frac{95}{49}x_1^2x_2^{-1})^c(\frac{95}{49}x_2^5x_1^{-3})^d = \frac{-95}{49}x_1^5x_2^{-3}$$

This sets up an easily solvable systems of equations, and we have:

$$A = (\frac{-95}{49})^{\frac{3}{7}}$$

and

$$B = (\frac{-95}{49})^{\frac{-8}{7}}$$

Also,

$$a = \frac{1}{7}$$

$$b = \frac{3}{7}$$

$$c = \frac{16}{7}$$

$$d = \frac{-1}{7}$$

With these calculations, we have transformed

$$f(x_1, x_2) := x_1^5 - \frac{49}{95}x_1^3x_2 + x_2^6$$

$$g(x_1, x_2) := x_2^5 - \frac{49}{95}x_2^3x_1 + x_1^6$$

into

$$r(y_1, y_2) := y_1 - 1 + y_2$$

$$s(y_1, y_2) := 1 + Ay_1^ay_2^b + By_1^cy_2^d$$

For the purpose of this project, systems will not be this difficult, but this example shows that any binomial can be transformed into the case of the monic binomial to which our algorithm solves. So, given any system not in the form of the algorithm input, we would simply add rescaling and a change of variable to the tasks to be completed before the initiation of the code.

## 3.2 Real Roots and the Bisection Method

With this monic binomial, we can define the precise precisions or real roots and positive roots as follows:

Suppose now that $d \in \mathbb{Z}$ and $c \in \mathbb{R}$.

**Find the precise conditions on $(c, d)$ under which $x^d = c$ has a positive root.**

1. When $d = 0$, we have that $x^0 = 1 = c$. So, $x^d = c$ has a positive root under $(1, 0)$.
2. When $d \neq 0$, we have that $x^d = c \Rightarrow x = c^{\frac{1}{d}}$. For x be positive, c must be positive. If c is negative we could have a complex or negative root. Note that we can't compare complex numbers so we have to exclude them.
3. If $c = 0$, we have that x must be zero. But this isn't a positive root.
4. The set $S$ under which $x^d = c$ has a positive root is $S = \{(c, d)/c > 0 \text{ and } d \neq 0\} \cup \{(1, 0)\}$

**Find precise conditions on $(c, d)$ under which $x^d = c$ has a real root.**

1. When $d = 0$, we have that $x^0 = 1 = c$. So, $x^d = c$ has a real root under $(1, 0)$.
2. When $d \neq 0$ we can consider $x = c^{\frac{1}{d}}$ and we have three cases: if $c > 0$, if $c = 0$ and if $c < 0$.
   (a) With $c > 0$, for any nonzero d we have a positive-real root.
   (b) With $c = 0$, x must be zero for any nonzero d and also this it a real root.
   (c) In the third case, $c < 0$, consider two cases for nonzero d: d is odd or d is even. When c is negative and d is an even number, $x^d = c$ has a complex root. When c is negative and d is an odd number, $x^d = c$ has a negative-real root.
3. The set $S$ under which $x^d = c$ has a real root is $S = \{(c, d)/c \geq 0 \text{ and } d \neq 0\} \cup \{(c, d)/c < 0 \text{ and d is an odd number}\} \cup \{(1, 0)\}$

So with this information, let us consider the monic binomial, Once again, we will present work and then follow with explanations.

Lemma: $x^d - c$ where $c > 0$, $c \in \mathbb{R}$, $d \in \mathbb{N}$, and $d \geq 2$ is a solution to Smale's 17th problem.

$$\text{Consider } f(x) := x^d - c$$

1. $c = 1$ then $x = 1$ DONE.

2. $c > 1$ Bisection method $\left|x_n - |c|^{\frac{1}{d}}\right| < \frac{1}{3d} < \frac{3-\sqrt{7}}{d-1}|c|^{\frac{1}{d}}$ Using bisection method on interval $[0, c]$ Then $n = \log(c) - \log(\frac{1}{3d}) = \log(c) - \log(3d)$ and the number of steps in this bisection process is $O(\log(c) + \log(d))$ and we must factor in the complexity of $f(x_n) = x_n^d - c$ which can be bounded (upper) at $O((log(d)^2))$, so the average complexity for this case is $O((log(d)^2))(\log(c) + \log(d))$

3. $c < 1$ choose interval $[0, 1]$ such that $f(0) = -c < 0$ and $f(1) = 1 - c > 0$ Therefore, a root exists in the interval according to the IVT. So, using the bisection method $\left|x_n - |c|^{\frac{1}{d}}\right| < \frac{3-\sqrt{7}}{d}|c| < \frac{3-\sqrt{7}}{d-1}|c|^{\frac{1}{d}}$ and the average complexity for this case is $O((\log(d)^2))(\log(d) - \log(c))$

   We therefore show that the average complexity of this case is $O((\log(d)^2))(\log(d) \pm \log(c))$. In this result based on this method, our algorithm will run in logarithmic time, which actually exceeds the complexity goal (for this part of the algorithm).

This project focused only on the real roots to systems of binomial equations, which means that cases of complex roots. The concept of a defined epsilon is very important for this research because we deal exclusively with root approximation, not exact solutions. We must therefore tell our bisection method algorithm what an approximate root is numerically so that it may stay in the bounds of mathematical accuracy. These bounds (as seen above in the Lemma) follow from the gamma theorem of Smale [3] which follows:

Suppose $f : \mathbb{C}^n \Rightarrow \mathbb{C}^n$ is analytic... If $|z-\zeta| \leq \frac{3-\sqrt{7}}{2} \times \frac{1}{\gamma(f,z)}$ then $z$ is an approximate root with true root $\zeta$
Gamma theory:

$$\gamma(f,z) = sup_{k\geq 2}\left|\frac{f'(z)^{-1}f^k(z)}{k!}\right|^{\frac{1}{k-1}}$$

Then for $f(x);= x^d - c$ where $d \in \mathbb{N}$ and $c \in \mathbb{C}^*$

$$f'(x) = dx^{d-1}$$

$$f^k(x) = d(d-1)\dots(d-(k+1))x^{d-k}$$

$$\gamma(f,z) = \left|\frac{d-1}{2z}\right|$$

So, if $\zeta$ is a true root of $x^d - c$, then $z$ satisfies

$$|z-\zeta| \leq \frac{3-\sqrt{7}}{d-1} \times |z|$$

This bound gives us an exact number for $\epsilon$. Note that this calculation took place for $c \in \mathbb{C}^*$, and the complex numbers contain the real numbers, so this bound holds for real $c$. Also in the algorithm, we have $c \in \mathbb{R}_+$ to ensure real solutions.

### 3.3 Matrix Exponentiation and Smith Normal Form

We will now explain the notation $x^A = c$, which was heavily used in this research. Given any real $nxn$ matrix M=$[m_{ij}]$ and $x \in \mathbb{R}^n_+$, let us define (* denotes multiplication)

$$x^M := (x_1^{M_{11}}...x_n^{M_{n1}}, ..., x_1^{M_{1n}}...x_n^{M_{nn}}).$$

This function has interesting properties that are easy to prove. The first is as follows:
Show that $x^{MB} = (x^M)^B$ , for any real $nxn$ matrix $B$.

$$[MB]_{ik} = \sum_{l=1}^{n} m_{il} * b_{lk} \tag{1}$$

for the jth term of $[x^{MB}]$,

$$[x^{MB}]_j = (x_1^{[MB]_{1j}} * x_2^{[MB]_{2j}} * ...x_n^{[MB]_{nj}}) \tag{2}$$

$$= (x_1^{\sum_{l=1}^n m_{1l}*b_{lj}} * x_2^{\sum_{l=1}^n m_{2l}*b_{lj}} * ... * x_n^{\sum_{l=1}^n m_{nl}*b_{lj}})$$

Then, for the jth term of $[x^M]^B$

$$[x^M]^B_j = (x_1^{M_{11}} * x_2^{M_{21}} * ...x_n^{M_{n1}})^{b_{1j}} * ... * (x_1^{M_{1n}} * x_2^{M_{2n}} * ... * x_n^{M_{nn}})^{b_{nj}} \tag{3}$$

$$= (x_1^{M_{11}*B_{1j}} * x_1^{M_{12}*B_{2j}} * ... * x_1^{M_{1n}*B_{nj}}) * ... * (x_n^{M_{n1}*B_{1j}} * ...x_n^{M_{nn}*B_{nj}})$$

$$= (x_1^{\sum_{l=1}^{n} m_{1l}*b_{lj}} * x_2^{\sum_{l=1}^{n} m_{2l}*b_{lj}} * ... * x_n^{\sum_{l=1}^{n} m_{nl}*b_{lj}})$$

Therefore $x^{MB} = (x^M)^B$ Also we can show that this function maps well from one finite field into the same field, and that matrices with det $\neq 0$ allow $x^A$ to be inverted in the same field. The proof follows:

(b)Prove that $f(x) := x^M$ is invertible ($\mathbb{R}_+^n$ to $\mathbb{R}_+^n$) $\iff$ det $M \neq 0$

$(i)$. Assume det $M \neq 0$, then show det $M \neq 0 \Rightarrow f(x)$ is invertible. For any real $nxn$ matrix $[M]$ whose det $\neq 0$, there exists matrix $[N]$, such that $M * N = \mathbb{I} = N * M$ , namely $M^{-1}$

Define $g(x)$ such that $g(f(x)) = x$ for $f(x) = x^M$. Then $g(x) = x^N$ so that $g(f(x)) = (x^M)^N = (x^{MN})$ by part a $= x^{\mathbb{I}}$

Check that $x^{\mathbb{I}} = x$
For $\mathbb{I}$ all entries are zero except $\mathbb{I}_{ll}$ where $l = 1, 2, 3...n$),
Then the first term of $x^{\mathbb{I}} = (x_1^{\mathbb{I}_{11}} * x_2^{\mathbb{I}_{12}} * ... * x_n^{\mathbb{I}_{1n}}) = (x_1^1 * x_2^0 * ... * x_n^0) = x_1$
and the nth term of $x^{\mathbb{I}} = (x_1^{\mathbb{I}_{1n}} * x_2^{\mathbb{I}_{2n}} * ... * x_n^{\mathbb{I}_{nn}} = (x_1^0 * x_2^0 * ... * x_n^1) = x_n$ Therefore $x^{\mathbb{I}} = x$ and $g(x) = f^{-1}(x)$ and with the inverse defined, det $M \neq 0 \Rightarrow f(x)$ is invertible

$(ii)$. Assume $f(x)$ is invertible, show that the det $M \neq 0 \equiv f(x)$ invertible

$\Rightarrow$ det $M \neq 0$
Proof by contrapositive: det $M = 0 \Rightarrow f(x)$ is not invertible $\equiv$ det $M = 0 \Rightarrow f(x)$ is not injective (injectivity required for inverse)
Since det $M = 0$, then $xM = 0$ has nontrivial solutions. Show $f(x)$ is not injective.
Let $\alpha = (\alpha_1...\alpha_n)$ be a non-zero vector in the left null space of M. Then $\alpha M = 0$
And $(\alpha_1...\alpha_n) \times \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ M_{n1} & \cdots & \cdots & M_{nn} \end{pmatrix} = \begin{pmatrix} \sum_{l=1}^{n} \alpha_l \times M_{l1} \\ \cdots \\ \sum_{l=1}^{n} \alpha_l \times M_{ln} \end{pmatrix} = \begin{pmatrix} 0 \\ \cdots \\ 0 \end{pmatrix}$ Then, let $W = \begin{pmatrix} 1 \\ \cdots \\ 1 \end{pmatrix}$ and $Y = \begin{pmatrix} 2^{\alpha_1} \\ 2^{\alpha_2} \\ \cdots \\ 2^{\alpha_n} \end{pmatrix}$ which are two distict vectors given $[\alpha]$ is nontrivial

Then $f(Y) = Y^M = \begin{pmatrix} 2^{\alpha_1 \times M_{11}} \times \cdots \times 2^{\alpha_n \times M_{n1}} \\ \cdots \\ 2^{\alpha_1} \times M_{1n} \times \cdots \times 2^{\alpha_n} \times M_{nn} \end{pmatrix} =$

$$\begin{pmatrix} 2^{\sum_{l=1}^{n} \alpha_l \times M_{l1}} \\ \cdots \\ 2^{\sum_{l=1}^{n} \alpha_l \times M_{ln}} \end{pmatrix} = \begin{pmatrix} 2^0 \\ \cdots \\ 2^0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{pmatrix}$$ according to the previous step

The same summation will hold true for any constant $c \in \mathbb{Z}$, and since W con-

tained only 1 values, it also yields $\begin{pmatrix} 1 \\ 1 \\ \dots \\ 1 \end{pmatrix}$.

Therefore $f(W) = f(Y)$, but $W \neq Y$. Therfore $f(x)$ is not injective
Since $f(x)$ is not injective, $\det M = 0 \Rightarrow f(x)$ is not injective $\equiv f(x)$ is invertible$\Rightarrow \det M \neq 0$
Conclusion: $f(x) := x^M$ is invertible ($\mathbb{R}^n_+$ to $\mathbb{R}^n_+$) $\iff \det M \neq 0$

The Smith Normal Form of an integer matrix is given in form $UAV = S$ where $S$ is a positive, diagonal matrix. Note: we only work with square systems that produce $n \times n$ matrices. In 1979, Kannan and Bachem [1] provided information to supplement Smith in his work on diagonal matrix by providing and algorithm to quickly calculate the SNF. The paper is cited, and we direct the reader to this for more information on integer matrix utilities and their usefulness in applied mathematics. Now we will briefly give an example of how the Smith Normal Form was utilized in our example with a small $2 \times 2$ matrix example. The example also shows how the concept of a monomial change of variable ($y^U = x$ and $\tilde{y}^U = \tilde{x}$) is used in our algorithm.

Given

$$x_1^2 x_2^4 = 5$$

$$x_1^6 x_2^8 = 9$$

$$UAV = S$$

$$\begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix} \begin{pmatrix} 2 & 6 \\ 4 & 8 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

$$(x^A = c) \Rightarrow (y^U = x) \Rightarrow (y^{UA} = c) \Rightarrow (y^{UAV} = y^S = c^V)$$

Then

$$(y_1)^2 = 5$$

$$(y_2)^4 = \frac{9}{5}$$

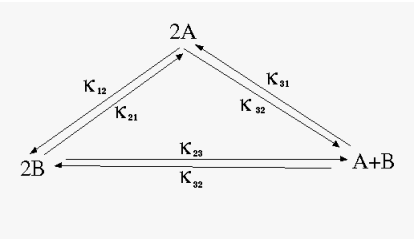
These Univariate binomials will form the vector $\tilde{y} = (\sqrt{5}, \sqrt[4]{\frac{9}{5}})$. This would be solved in our algorithm with error $\epsilon$ by $f = y_1^2 - 5$ and $g = y_2^4 - \frac{9}{5}$. Then the vector $\tilde{y}$ is raised to matrix $U$ to produce approximate root vector $\tilde{x}$ which will be shown to fall in the bound for the approximate root definition. Therefore we can summarize this process as:

SNF $\Rightarrow$ Univariate binomials $\Rightarrow$ Previous algorithm $\Rightarrow \tilde{y} \Rightarrow \tilde{y}^U = \tilde{x}$.

## 3.4 Chemical Reaction Networks

In 2011, Millán et. al provided the paper "Chemical Reaction Systems With Toric Steady States" [2], which provide sufficient conditions for a Chemical Reaction Network (CRN) to have a system of ODE's expressible as a binomial system. A **Chemical Reaction Network (CRN)** is an organized digraph of chemical reactions. Under the assumption of *mass-action kinetics*, chemical species react at a rate proportional to the product of their concentrations, where the proportionality constant is the rate constant $\kappa$. In chemistry, a steady state is a situation in which all state variables are constant in spite of ongoing processes that strive to change them. It is called a **toric steady state** if the corresponding system of ODE's can be expressed as a binomial system and then solved.We know how to scale any binomial system into a system of monic binomials, so we can apply our algorithm to real bio-models with defined parameters. Here is an example of how a CRN is expressed as differential equations and made into a binomial system:

**Reaction** (Triangle Network): Let $s = 2$ and $m = 3$. Then $2A = x_1^2$, $2B = x_2^2$, and $A + B = x_1x_2$




$$\frac{dx_1}{dt} = -\frac{dx_2}{dt} = (-2\kappa_{12} - \kappa_{13})x_1^2 + (2\kappa_{21} + \kappa_{23})x_2^2 + (\kappa_{31} - \kappa_{32})x_1x_2$$

As only the coefficients of $x_1x_2$ can be zero, this system has toric steady states iff $\kappa_{31} = \kappa_{32}$.

As previously stated, we have precise conditions under which a CRN can be expressed as a binomial system. In the paper [2], these are conditions 3.1, 3.4, and 3.6. They are all essentially linear algebra conditions. Unfortunately in this small research period, there are no finite results with the bio-models we found in an on-line database. We are very close, using Computer Algebra System *Maple*. These linear algebra conditions apply to a $27 \times 27$ matrix for our bio-model, so calculations with Gaussian elimination for the right null-space will only take a few more days of research. If anyone is interested in the Maple file, the researcher will gladly provide it.

# 4 Future Work

Therefore, we can:

1. Find biomodels with defined paramaters ($\kappa$ denotes reaction constants)
2. Convert these models' associated dynamical systems into binomial systems based on the work of Millán et. al
3. Use binomial system as input for developed binomial system algorithm

The solutions to such systems are of high importance to Computational & Systems Biologists. The head undergraduate researcher for this project would enjoy the opportunity to further pursue results with these mathematical biologists. Also, we find it interesting to find applications of binomial system solving in areas such as Business Statistics and Economics. This project is very much open to many academic disciplines, and the undergraduate researcher has future aspirations to serve as a bridge between mathematicians and experts in other fields.

# 5 Algorithm

Below is the Sage code that was written for this project to solve binomial systems. This particular code is for a $(2 \times 2)$ system. For larger systems, $c$ has as many points as columns in $A$ More general code can be received from the researcher for anyone interested:

```
def vector_exp(v):
    m=len(v)
    R=PolynomialRing(RR,m,"x")
    x=R.gens()
    comp=prod(x[i]^v[i] for i in range(m))
    return comp

def matrix_exp(x,M):
    n = M.ncols()
    M = transpose(M)
    result = [vector_exp(M[i]) for i in range(n)]
    return result

def yvector_exp(v):
    m=len(v)
    y=var('y')
    R=PolynomialRing(RR,m,"y")
    y=R.gens()
    comp=prod(y[i]^v[i] for i in range(m))
    return comp

def ymatrix_exp(y,M):
    n = M.ncols()
    M = transpose(M)
    result = [yvector_exp(M[i]) for i in range(n)]
    return result

def bisect_method(f, a, b, eps):

    try:
        f = f._fast_float_(f.variables()[0])
        except AttributeError:
            pass
    intervals = [(a,b)]
    two = float(2); eps = float(eps)
    while True:

        c = (a+b)/two
        fa = f(a); fb = f(b); fc = f(c)
```

```
        if abs(fc) < eps: return c, intervals
        if fa*fc < 0:
            a, b = a, c
        elif fc*fb < 0:
            a, b = c, b
            else:
                    raise ValueError, "f must have a sign change in the interval (%s,%s)"%(a
            intervals.append((a,b))

def _(f = cos(x) - x, a = float(0), b = float(1), eps=(-3,(-16..-1))):
     eps = 10^eps
     print "eps = %s"%float(eps)
     try:
         time c, intervals = bisect_method(f, a, b, eps)
     except ValueError:
         print "f must have opposite sign at the endpoints of the interval"
         show(plot(f, a, b, color='red'), xmin=a, xmax=b)
     else:
         print "root =", c
         print "f(c) = %r"%f(c)
         print "iterations =", len(intervals)
         P = plot(f, a, b, color='red')
         h = (P.ymax() - P.ymin())/ (1.5*len(intervals))
         L = sum(line([(c,h*i), (d,h*i)]) for i, (c,d) in enumerate(intervals) )
         L += sum(line([(c,h*i-h/4), (c,h*i+h/4)]) for i, (c,d) in enumerate(intervals) )
         L += sum(line([(d,h*i-h/4), (d,h*i+h/4)]) for i, (c,d) in enumerate(intervals) )
         show(P + L, xmin=a, xmax=b)



def single_binomial(d, c):

    c = float(c); d = int(d)
    x = var('x')
    f = x^d-c
    eps = .0001

    if c > 1:
        return bisect_method(f, 0, c, eps)
    elif c < 1:
        return bisect_method(f, 0, 1, eps)
    elif c == 1:
        raise ValueError, "x is equal to one"

def S_diag (A):
```

```
    S, U, V = A.smith_form()

    return S

def U_diag (A):

    S, U, V = A.smith_form()

    return U

def V_diag (A):

    S, U, V = A.smith_form()

    return V

def SNF_diag(A):

    S= S_diag(A)
    y = var('y')
    r = A.ncols()
    return ymatrix_exp(y,S)


def binomial_system(A,c1, c2):

    c1 = int(c1)
    c2 = int(c2)
    S = S_diag (A)
    B = single_binomial(S[0][0], c1)
    C= single_binomial(S[1][1], c2)

    return B, C
```

# 6 References and Acknowledgments

# References

[1] Kannan & Bachem. Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix. *Society for Industrial and Applied Mathematics, 1979.*

[2] Millán, Dickenstein, Shiu, & Conradi Chemical Reaction Systems With Toric Steady States *arXiv:1102.1590 [math.DS]*, 2011.

[3] Smale, Steve Newton's Method Estimates from Data at One Point *The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics, 1986.*

**Acknowledgements**

Kaitlyn Phillipson, *Lead Mentor*
Dr. J. Maurice Rojas, *Lead Professor*
The National Science Foundation (NSF)
Texas A&M University Peers, Faculty, and Friends of the Texas A&M Mathematics REU 2014

I would like to thank all those who have supported me in my research, especially those mentioned above. This research experience was stimulating and full of new experiences. My future as a researcher and business professional will always be impacted by the experience I shared with the 13 other students in this program.