On Initial Conditions for Biological Simulations

Cristina Pop

Project Number: MO2
Supervisor: Dr. Meeko Oishi

Electrical and Computer Engineering 496
University of British Columbia
April 11, 2007
ABSTRACT

“Initial Conditions for Biological Simulations”

by

Cristina Pop

This report investigates how computational tools can be used to model and simulate biological processes described by hybrid systems. Specifically, the parameters and the initial conditions that lead the system to a target state are determined for tryptophan regulation. Fast, accurate, and simultaneous identification of these two characteristics pose great benefit to biological experiments by allowing scientists to generate a large amount of inexpensive computational tests to solidify their hypotheses and justify further wet-lab experimentation.

The Level Set Toolbox and the RAMAS software package were analyzed as representative reachability and verification techniques available: a direct method, reachability tool and an indirect method, model checking tool. The Level Set Toolbox was a reliable but slow tool for determining initial conditions. Although it can produce an exact solution for any type of system dynamics, it could not be used to determine the parameters for tryptophan regulation because the increased dimension of the new system would unreasonably increase the running time. RAMAS was more difficult to set up; a short program was created to efficiently represent the dynamics of large systems and automate the input procedure. A complete set of initial conditions was not determined using RAMAS because it would have been necessary to verify all possible points in the domain. Instead, a set of initial conditions that did not lead all trajectories to the target set was obtained. The results were in agreement with those of a higher-dimensional system that included the unknown parameters. These initial tests show that RAMAS is a useful and fast tool for large systems that the Level Set Toolbox cannot handle. Using the series of tests presented in this paper, it is also possible to explore the robustness of a particular parameter and determine the threshold value that leads the system outside the target set.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... ii
List of Illustrations ......................................................................................................... iv
Glossary .......................................................................................................................... v

1.0 Introduction ................................................................................................................ 1

2.0 Hybrid Systems and Verification ............................................................................. 4
  2.1 Hybrid Systems ......................................................................................................... 4
  2.2 Hybrid System Verification ..................................................................................... 6
  2.3 Level Set Toolbox ................................................................................................... 9
  2.4 RAMAS .................................................................................................................. 11

3.0 Biological Models ...................................................................................................... 16
  3.1 Tryptophan Regulation .......................................................................................... 16
  3.2 Gene Transcription in *Vibrio fischeri* ................................................................. 22

4.0 Computational Experiments ..................................................................................... 23
  4.1 *Vibrio fischeri* Test Sets ..................................................................................... 23
    4.1.1 Level Set Toolbox Tests ................................................................................ 23
    4.1.2 RAMAS Tests ............................................................................................... 27
  4.2 Tryptophan Test Sets ............................................................................................ 31

5.0 Results and Recommendations ............................................................................... 40

6.0 Conclusion ................................................................................................................ 43

References .................................................................................................................... 45
Appendix ......................................................................................................................... 47
LIST OF ILLUSTRATIONS

FIGURES

Figure 1: Initial Conditions Problem ................................................................. 2
Figure 2: Hybrid System Components ............................................................... 4
Figure 3: Partitioning of a Continuous State Space (a) into Discrete Quotients (b) .... 12
Figure 4: Tryptophan Production (a) and Repression (b) ................................. 17
Figure 5: Tryptophan Regulation as a Hybrid System ....................................... 18
Figure 6: V. fischeri Model on Level Set Toolbox – Initial Conditions for Target Set 1 .. 25
Figure 7: V. fischeri Model on Level Set Toolbox – Initial Conditions for Target Set 2 .. 26
Figure 8: V. fischeri Model on Level Set Toolbox – Initial Conditions for Target Set 3 .. 26

TABLES

Table 1: Classification of Automata and Reachability or Model Checking Tools ........ 7
Table 2: Level Set Toolbox Variables ................................................................. 11
Table 3: Metabolites Involved in Tryptophan Biosynthesis ............................... 18
Table 4: Known and Unknown Parameters in the Tryptophan Regulation Model .... 19
Table 5: V. fischeri Model on Level Set Toolbox ................................................ 24
Table 6: V. fischeri Model on RAMAS – First Test Set ..................................... 27
Table 7: V. fischeri Model on RAMAS – Second Test Set .................................. 28
Table 8: V. fischeri Model on RAMAS – Third Test Set ..................................... 30
Table 9: Dimension 5 Tryptophan Model on RAMAS – Initial Conditions Leading
Outside Target .................................................................................................... 32
Table 10: Higher Dimension Tryptophan Model on RAMAS ............................ 33
Table 11: Dimension 6 Tryptophan Model on RAMAS ..................................... 34
Table 12: Dimension 7 Tryptophan Model on RAMAS ..................................... 36
Table 13: Running Time Analysis for RAMAS .................................................... 38
Table 14: Vector Field Coefficients for V. fischeri Uncontrolled Vector Field Model on
RAMAS ............................................................................................................ 47
## GLOSSARY

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>amino acid</td>
<td>an organic molecule that serves as a building block of proteins</td>
</tr>
<tr>
<td>chromosome</td>
<td>a long strand of DNA containing the hereditary information of an organism</td>
</tr>
<tr>
<td>operator</td>
<td>a short region of DNA that is recognized by a repressor protein and thereby controls transcription of a neighbouring gene [1]</td>
</tr>
<tr>
<td>RNA polymerase</td>
<td>an enzyme that initiates transcription based on recognition of an identifying element on a DNA strand [1]</td>
</tr>
<tr>
<td>transcription</td>
<td>the process through which a DNA strand is copied into a complementary RNA strand; initiated by RNA polymerase [1]</td>
</tr>
<tr>
<td>tryptophan repressor</td>
<td>a gene regulatory protein for tryptophan production, which binds to an operator [1]</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

Biological processes within the cell involve feedback mechanisms and complicated pathways that biologists are only now starting to understand. Modeling and simulation of biological processes is therefore useful in obtaining a thorough analysis of the governing mechanisms of biological networks. This project investigates how emerging computational tools in hybrid control theory can be used to provide accurate modeling and simulation of these processes. Since hybrid systems hold information on both discrete behaviour as well as smooth, dynamical behaviour, they can characterize such biological networks that exhibit both attributes. With growing work in the field, hybrid systems tools can guide biologists in determining critical patterns in biological processes.

Protein regulation is a particular example of a biological process. In protein regulation, the concentrations of one or more molecules grow according to one flow equation until a certain threshold is reached. At that point, a different type of growth or recession is followed. This process can be described as a system that switches between different discrete states governed by their own continuous dynamics: a hybrid system. This project will consider the process of tryptophan regulation, whereby production of the amino acid tryptophan is turned on and off as a function of the level of tryptophan in the cell.

Simulation of tryptophan regulation is hindered by potentially unknown or inaccurate model parameters and unknown initial conditions that draw the system toward a specific outcome. However, the initial conditions and the parameters can both be computed from a known dynamical model of tryptophan regulation. The objectives of this project are to:

1. Determine the initial conditions for tryptophan regulation when the parameters are known.
2. Simultaneously determine the parameters and initial conditions when both are unknown.
Figure 1 illustrates the initial conditions problem (Objective 1), which can be solved through two different types of verification techniques: reachability and model checking.

![Diagram of initial conditions problem]

**Figure 1: Initial Conditions Problem**

Reachability tools determine all possible initial states that drive the system to a target state, while model checking tools verify whether a target state is reachable from a given initial state. In the case of tryptophan regulation, the model consists of state variables representing concentrations of the reactants and products involved in the chemical reactions governing the process. Given a final set of values for each state variable, the goal is to determine all possible initial values of these variables.

To solve the two objectives, the biological tryptophan regulation process and its mathematical model will be investigated. Both model checking tools and reachability tools will be applied to the model, and compared in terms of speed, accuracy, generality (in terms of the types of mathematical models supported), and ease of use.

Developing an efficient method of obtaining a complete model – consisting of viable parameters and initial conditions – can help biologists test their hypotheses in a simulation environment, without wasting expensive laboratory resources. Knowledge of what starting points lead to a target state can aid in determining reactant input values and
how the system will behave under various influences. Knowledge of the parameters in a biological model can also give insight into reaction rates in chemical equations.

Many tools have been developed to verify discrete and continuous systems, but fewer tools have been created for verification of hybrid systems. Further, reachability tools are limited in the types of systems they handle or offer fast computation times only for models of a certain form [2]. No computational tools have been developed for explicitly solving the parameter and the initial conditions problem simultaneously. The main significance of this work lies in the analysis of two complementary tools that offer different benefits and pose different limitations, and in the application of these tools to solve the problem objectives. In addition, the observations and results generated throughout the project can be applied to other biological models that are of interest to applications in therapeutics, human health, energy sources, defense systems, and other research areas. Obtaining a good understanding of the techniques available allows us to better choose the right tool for solving a specific problem with certain restrictions on timeliness and validity of results.

The work performed during this project was divided into two major components: research of the tryptophan regulation model and investigation of computational tools that can be used to solve the objectives. Because this project is oriented toward the computational aspect, there is no derivation of the biological model shown in this report. Rather, a detailed analysis of each computational tool is presented. The programming languages used in this work are Matlab and C++. Due to timing constraints, not all tests of interest were performed. A summary of some important tests and their significance to the usefulness of each tool will be described. Several ways to extend the project will be also presented based on the initial set of results obtained for the tryptophan model. This report divides into the following primary sections: Hybrid Systems and Verification, Biological Models, Computational Experiments, Results and Recommendations, and Conclusion.
2.0 HYBRID SYSTEMS AND VERIFICATION

Hybrid systems are ideal for describing biological processes with both discrete and continuous components. The reachability properties of such systems are important issues that many computational tools have aimed to solve. In this section, we describe hybrid systems in the context of biological processes and explore the types of verification tools that have been developed.

2.1 Hybrid Systems

Hybrid systems have two components: a discrete component and a continuous component. Figure 2 shows these two components for tryptophan regulation.

![Figure 2: Hybrid System Components](image-url)
In tryptophan regulation, the system switches between two discrete states: one in which tryptophan production is on and one in which production is off. The biological process involved will be described further in Section 3.1. Each discrete state has its own continuous dynamics. These dynamics are represented by a system of differential equations that together describe how certain representative variables (the state variables $x_i$) change over time. In a biological process, a state variable could be the concentration of a protein in the cell.

The notation and nomenclature used in this paper are as follows. The dimension of the system is defined to be the number of state variables, denoted by $n$ in this paper. A trajectory is defined to be the evolution of a state variable in time. The function $f(x, u, d)$ represents the continuous dynamics in one discrete mode (i.e. a system of differential equations $x_i = f_i(x)$ for each state variable $x_i$, where $i = 1, 2, \ldots, n$). Note that when we refer to $x$ in the context of the dynamics of a system, we mean the vector $x$, which is defined as $[x_1, x_2, \ldots, x_n]$. In a complete model, the function $f$ depends not only on the state variables, but also on a vector of control inputs $u$ defined on a set $U$ and a vector of disturbance inputs $d$ defined on a set $D$. The control inputs are variables in the system that can be controlled. For example, a control input might represent an external signal originating from another system that regulates the behaviour of the current system [3]. The disturbance inputs are unknown variables that cannot be controlled. The inputs originate either from within the current system or from the actions of another system in the environment [2]. In cellular systems, for example, perturbations can occur due to external influences such as variable environment characteristics or internal mutations that change the structure of the system, eliminate substrates necessary in a reaction, or modify the kinetic properties of molecules [3].

Many examples of hybrid systems exist in the field of systems biology. Signal transduction networks involve feedback mechanisms similar to tryptophan regulation [3]. Planar cell polarity signaling in the wing epithelium of a fruit fly is another example. Planar cell polarity is an intracellular process through which proteins that control the
polarity of hair growth on a fly’s wing localize to different cell areas [4]. Delta-Notch protein signaling is another intercellular signaling mechanism through which the two transmembrane proteins Delta and Notch cause growth and recession of Delta and Notch proteins in neighbouring cells. This signaling mechanism causes the formation of planar cell polarity and other patterns of cells and biological characteristics [5]. The discrete states of each process and the unique dynamics that occur within that state form the hybrid system.

2.2 Hybrid System Verification

Consider Figure 1, which depicts the problem of hybrid system verification. Several approaches to verification exist. A summary of the most developed approaches is shown in Table 1. The table is organized according to the system dynamics, an important characteristic determined by the set of differential equations \( f(x) \) in each mode. Also included in the table is the type of tool (reachability or model checking, as described in the Introduction) and, if available, whether the method used by the tool is direct or indirect (as defined later in this section). The language each tool was implemented in is also included if available. Note that this is not an exhaustive list or available tools or possible system dynamics. An open compilation of other hybrid systems tools is available at [6].
<table>
<thead>
<tr>
<th>Automata (Dynamics)</th>
<th>Model</th>
<th>Description / Comments</th>
<th>Tools</th>
<th>Type of Tool</th>
<th>Direct/Indirect</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed</td>
<td>$f(x) = 1$</td>
<td>each $x_i$ is a clock which increments uniformly over time</td>
<td>Uppaal [7]</td>
<td>model checking</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kronos [8]</td>
<td>model checking</td>
<td></td>
<td>Java</td>
</tr>
<tr>
<td>Rectangular</td>
<td>$f_i(x) \in [a,b]$</td>
<td>most tools start with a complex version of the equations and simplify it to a rectangular automata</td>
<td>HyTech [9]</td>
<td>model checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kronos [8]</td>
<td>model checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Uppaal [7]</td>
<td>model checking</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>$f(x) = Ax$</td>
<td>most tools start with a complex version of the equations and simplify it to a rectangular automata</td>
<td>Multi-Perimetric Toolbox [10]</td>
<td>reachability and verification</td>
<td>Matlab</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MATISSE [12]</td>
<td>model checking</td>
<td>indirect</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HyTech [9]</td>
<td>model checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d/dt [13]</td>
<td>reachability and model checking</td>
<td>direct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Checkmate [14]</td>
<td>model checking</td>
<td>direct</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Charon [15],[16]</td>
<td>model checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Affine</td>
<td>$f(x) = Ax + b$</td>
<td></td>
<td>Multi-Perimetric Toolbox [10]</td>
<td>reachability and verification</td>
<td>direct</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d/dt [13]</td>
<td>reachability and model checking</td>
<td>direct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Checkmate [14]</td>
<td>model checking</td>
<td>direct</td>
<td>Matlab</td>
</tr>
<tr>
<td>Multi-affine</td>
<td>$f(x) = \sum_{i,j \in {1,...,n}} c_{ij} x_i x_j$</td>
<td>The degree of each variable $x_k$ is not greater than 1. Equivalently, the function is affine in each variable $x_k$. [17]</td>
<td>Level Set Toolbox [18]</td>
<td>reachability</td>
<td>direct</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RAMAS [19]</td>
<td>model checking</td>
<td>indirect</td>
<td>Matlab</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RoVerGeNe [20]</td>
<td>model checking of parameter robustness and constraints (piece-wise multi-affine)</td>
<td>Matlab</td>
<td></td>
</tr>
</tbody>
</table>
In this paper, we will focus on multi-affine dynamics. Further, we will be investigating two tools for biological models with multi-affine dynamics: the Level Set Toolbox and RAMAS. These tools were chosen because, as shown below, they offer different perspectives on the types of tools and techniques available. First, we distinguish model checking tools (RAMAS) from reachability tools (Level Set Toolbox). Then we discuss direct and indirect methods.

With model checking tools, an initial condition is input into the tool and the tool computes whether or not the system, starting from those particular initial conditions, can reach the final set within a certain time period. This method is clearly simple to implement since it is known (from the differential equations), how each variable in the system behaves over time. The disadvantage of using a model checking technique is that to solve the initial conditions problem, all possible initial conditions within the domain must be checked to determine if the target set is reachable. Initial conditions that reach a point outside the target set do not satisfy the initial conditions problem and can be rejected.

Reachability tools determine the set of all possible initial conditions that could lead the system state to the target. This method is therefore harder to implement, since the system must be evolved backward in time from the target set, so that the start set can be determined. Two different techniques exist to compute reachable sets: direct and indirect methods.

Direct methods construct the reachable set without simplifying the continuous state space [17]. Because the state space is uncountable, this type of method is typically undecidable: it is unknown whether a solution exists. The Level Set Toolbox [18] is an example of a tool that uses such a technique, in which the boundary of the target set is evolved backward in time based on the dynamics of the system. If the set converges after a certain time period, the set of all initial conditions has been found.
Indirect methods successively partition the continuous state space so that a continuous or hybrid system is mapped to a discrete system. The goal is to create partitions of the original domain such that the new system simulates the dynamics of the original system on each partition. RAMAS [19] is an example of a tool that uses an indirect method. At each refinement of the state space, a discrete quotient is computed. The discrete quotient contains a set of trajectories that encompasses the trajectories of the original system. The partitioning method used by RAMAS may include trajectories in the discrete quotient that do not arise in the original system [17]. This is a conservative approach, which is one of the reasons that indirect methods generate approximate solutions to the reachability problem while direct methods yield exact solutions.

2.3 Level Set Toolbox

The dynamics of a biological process like tryptophan regulation can be described by a Hamilton-Jacobi equation, which is a first-order partial differential equation. Level set methods are numerical algorithms that can be used to solve Hamilton-Jacobi problems [18]. Using notation similar to [2], let $I(\tau)$ be the set of initial conditions that lead to the target set $T$. The target $T$ is a known set of $x$ values, where $x(t)$ is a trajectory of the system at time $t$. The set $I(\tau)$ is then defined as the set of $x(0)$ such that $x(s) \in T$ for $s \in [0, \tau]$.

More formally, we can specify a scalar function $\phi_0(x)$ defined over an $n$-dimensional space. That is, $\phi_0(x)$ is a function of $x_1, x_2, \ldots x_n$ and returns a single real value. The target value can then be described as:

$$T = \{x \in \mathbb{R}^n \mid \phi_0(x) \leq 0\}$$

With this terminology, $\phi(x)$ is a level set function, where the zero level set of this function represents the boundary of the set. The zero level set is a cost function: if it is
negative, the point \( x \) is within \( T \) and is a desired value; if it is positive, the point \( x \) is outside of \( T \) and is not a desired value; and if it is 0, the point is on the boundary. As the target set is evolved backwards in time, the desired areas will contract and the unwanted areas will grow. To determine \( I \), we have the following expression: 
\[
I(\tau) = \{x \in \mathbb{R}^n \mid \phi(x, -\tau) \leq 0\}
\]

The function \( \phi(x,t) \) obeys the following Hamilton-Jacobi equation:
\[
\frac{\partial \phi(x,t)}{\partial t} + \min \left\{ 0, H(x, \nabla \phi(x,t)) \right\} = 0
\]
\[
\phi(x,0) = \phi_0(x)
\]

where the various parameters are defined as follows:

- \( H(x, \nabla \phi(x,t)) = \max \{ \min \{ \nabla \phi(x,t) \cdot f(x,u,d) \} \} \) (i.e. the dot product of \( \phi \) and \( f \) minimized over disturbance inputs and maximized over control inputs, known as the Hamiltonian)
- \( \nabla \phi(x,t) = \left[ \frac{\partial \phi}{\partial x_1}, \frac{\partial \phi}{\partial x_2}, \ldots, \frac{\partial \phi}{\partial x_n} \right] \) (i.e. the gradient of \( \phi \))
- \( f(x,u,d) \) is defined as in the Section 2.1

As shown above, \( I(\tau) \) is determined by solving the Hamilton-Jacobi equation for \( \phi(x,t) \) backwards in time from 0 to \(-\tau\), or equivalently finding the zero level set of \( \phi(x,t) \) at time \( t = -\tau \). The derivation described above was based on [2] and [18]. The Level Set Toolbox evolves the boundary of the target set (the \textit{data0} variable) backwards in time to determine the set of initial conditions (the \textit{data} variable).

The Level Set Toolbox is written as a set of Matlab scripts that must be modified (i.e. there is no user interface). Table 2 shows the parameters that can be modified within the program and what they represent.
Table 2: Level Set Toolbox Variables

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>tMax</td>
<td>The value $\tau$ at which the simulation should end. This value would be optimized by the user by observing how quickly the results tend to converge over several runs of the program.</td>
</tr>
<tr>
<td>Nx</td>
<td>The number of grid points. Large values would yield more accurate results but within a longer time.</td>
</tr>
<tr>
<td>g.min = [a1, a2, ... an] g.max = [b1, b2, ... bn]</td>
<td>The minimum and maximum values of the domain ($a_1 &lt; x &lt; b_1$, etc).</td>
</tr>
<tr>
<td>hamValue</td>
<td>The Hamiltonian $H$ as defined above.</td>
</tr>
<tr>
<td>Alpha</td>
<td>A term that bounds how quickly the partial derivative of each state variable changes. This could simply be the absolute value of $f(x_i)$ for each $i = 1, 2, ..., n$.</td>
</tr>
<tr>
<td>data0 data</td>
<td>The initial and final sets as defined above.</td>
</tr>
</tbody>
</table>

Within the program, $data0$ and $data$ are initially set to the same value. After the program is finished running, $data$ holds values of the cost function $\phi(x,t)$ for each grid point at time $tMax$. If the dimension of the system is 2 or 3, the points of $data$ are also visualized on a graph.

The running time complexity of the Level Set Toolbox is $O(g^n)$ where $g$ is the number of grid points ($Nx$ in the table above). For systems with dimension greater than 5, the running time of the algorithm becomes unreasonably high [18].

2.4 RAMAS

RAMAS is a software package for analyzing multi-affine systems. Written as a set of Matlab script files, RAMAS is an example of the indirect method for determining reachable sets. It also acts as a model checking tool, since both the initial and final sets must be entered as inputs to the program. RAMAS verifies the system by considering the final set to be the complement of the target set (as opposed to the target itself). The tool
determines if the final set (called the stop set) is reachable from the initial set (called the start set). If the stop set – the complement of the target – is not reachable, then that particular initial condition does not lead to an undesired set. This means that it must lead to the desired target set and the initial condition can be accepted. If there are potentially reachable sets in the specified stop set, the initial condition values should be rejected (since there are some trajectories that do not lead to the target). In this last case, the output is a list of all sets reachable from the start set.

As described in Section 2.2, RAMAS is an indirect method, which partitions the continuous state space. The discrete quotient computed at each refinement step contains trajectories of the discrete system that originate and terminate within one of a finite number of regions of the continuous domain [17]. Figure 3 shows one refinement step for a sample system.

![Diagram](image)

**Figure 3: Partitioning of a Continuous State Space (a) into Discrete Quotients (b)**

Taken from: [17]

In part a.) of the figure, the trajectories of the continuous systems (the arrows) are shown. The domain is partitioned into regions (labeled $\text{con}(l)$) that include several trajectories of the continuous system. These regions are represented as a finite number of labels $l$ in the discrete quotient shown in part b.). In the discrete quotient, the trajectories shown in the figure (also represented as arrows) include trajectories that arise in the original system.
RAMAS has two restrictions: the program applies only to multi-affine dynamics and the program can only handle rectangular start and stop sets. The definition of multi-affine was given previously. To say that only rectangles are handled is to say that one of the following three equations holds for the start and stop sets [17]:

- \( x_i = a_i \) for \( i = 1, \ldots, n \) (0-order rectangle or a point)
- \( a_i \leq x_i \leq b_i \) or for \( i = 1, \ldots, n \) where \( a_i < b_i \) (full-dimension rectangle)
- \( x_i = b_i \) for \( i = 1, \ldots, n \) (0-order rectangle or a point)

The following is a sample template of the inputs provided to and the outputs produced by RAMAS. The labels in the template will be used in the next section to summarize the output obtained for various tests. Inputs are shown in bold and outputs are shown in bold italics.

**Dimension of space:** \( n \)

**Theta _ 1 vector** [\( \text{th}_0 \text{ th}_1 \ldots]: [t_1 \ a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ b_3 \ c_1 \ c_2 \ c_3 \ d_1 \ d_2 \ d_3 \ t_2] \)

**Theta _ 2 vector** [\( \text{th}_0 \text{ th}_1 \ldots]: [t_1 \ a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ b_3 \ c_1 \ c_2 \ c_3 \ d_1 \ d_2 \ d_3 \ t_2] \)

... \n
**Theta _ n vector** [\( \text{th}_0 \text{ th}_1 \ldots]: [t_1 \ a_1 \ a_2 \ a_3 \ b_1 \ b_2 \ b_3 \ c_1 \ c_2 \ c_3 \ d_1 \ d_2 \ d_3 \ t_2] \)

**Meaning:**
- \( a_1 < x_1 < b_1 \)
- \( a_2 < x_2 < b_2 \)
- \( a_3 < x_3 < b_3 \)

**Start set** (matrix 2 x 3 - [\( x_1 \min \ldots x_n \min; x_1 \max \ldots \]
\( x_n \max \], where \( \min \), \( \max \) are vectors formed with points from \( \text{theta}_i \), \( i=1,\ldots,3 \): [\( a_1 \ a_2 \ a_3; b_1 \ b_2 \ b_3 \])

**Stop set** (not desired to be reached from Start) - matrix 2 x 3: [\( c_1 \ c_2 \ c_3; d_1 \ d_2 \ d_3 \])

**Maximum resolution** (smallest distance between any theta points on any axis) ("0" to disable): \( r \)

**Maximum number of algorithm iterations** ("Inf" to disable): \( i \)

**Number of digits** to be taken into account in computations of vector field values and new theta points (in order to eliminate representation error): \( d \)

**Matrix** (3 x 8) containing vector field coeff. (column 1: free term; col 2: coef of \( x_1 \); col 3: coef of \( x_2 \); col 4: coef of \( x_2^*x_1 \); \ldots): **Model**
Iteration 1 -> $X$ rectangles
Labeling, rectangle order and limits ...
  time: $t_1$ secs
Signatures, transitions, adjacency matrix, ...
  time: $t_2$ secs
Reachable set ...
  time: $t_3$ secs

Property IS NOT satisfied!
Start region (closed): [a1 a2 a3] ... [b1 b2 b3]
Stop region (closed): [c1 c2 c3] ... [d1 d2 d3]

Open sets from stop region that probably can be reached from the start region: List

Property IS NOT satisfied!
Maximum resolution reached!
  Number of iterations: $ip$
  Resolution reached: $rp$
  Approximate total run time: $t_4$ secs

Note that there are $n$ lines generated for the theta vectors. Each theta vector is a list of the points that appear in the domain (the continuous state space) and the start and stop sets. No duplicate values are necessary. These points will be used to partition the state space. As the number of points increases, the running time of the algorithm may also increase. The start and stop sets format have the meaning shown. The number of digits $d$ is typically assumed to be 12. Typical values of resolution and number of iterations will be discussed further in Section 4.0. The format of the system dynamics (labeled as Model) is shown in the Appendix. The important outputs are:

- whether or not the stop set is reachable
- the number of rectangles used in the partitioning scheme, important for running time
- the total running time

Also shown in the template is the output produced when there are certain sets within the stop set that are reachable from the given start set. The List output presents these sets as rectangles in same format as the start and stop sets. If the stop set is not reachable from the start set (i.e. we have found a viable set of initial conditions that do not lead to the stop set and therefore must lead to the target), no List is produced and the output statement is “Property IS satisfied! Stop set is not reachable from start set.” An example input/output sequence follows:
>> RAMAS

Dimension of space: 3
Theta _1 vector [th_0 th_1 ...]: [0 1 2 20]
Theta _2 vector [th_0 th_1 ...]: [0 1 2 20]
Theta _3 vector [th_0 th_1 ...]: [0 1 2 20]

Start set (matrix 2 x 3 - [x1_min...xn_min;x1_max...xn_max], where min, max are vectors formed with points from theta_i, i=1,...,3): [1 1 1; 2 2 2]
Stop set (not desired to be reached from Start) - matrix 2 x 3: [0 0 0; 20 2 20]

Maximum resolution (smallest distance between any theta points on any axis) ('0' to disable): 2
Maximum number of algorithm iterations('Inf' to disable): 2
Number of digits to be taken into account in computations of vector field values and new theta points (in order to eliminate representation error): 12

Matrix (3 x 8) containing vector field coeff. (column 1: free term; col 2: coef of x1; col 3: coef of x2; col 4: coef of x2*x1; ...): [0 10 0 0 40 -40 0 0; 0 0 -10 0 0 30 0 0; 0 30 10 -10 10 -30 -10 10]

Iteration 1 -> 27 rectangles
   Labeling, rectangle order and limits ...
   time: 0.02 secs
   Signatures, transitions, adjacency matrix, ...
   time: 0.99 secs
   Reachable set ...
   time: 0.02 secs

Property IS NOT satisfied!
Start region (closed): [1 1 1] ... [2 2 2]
Stop region (closed): [0 0 0] ... [20 2 20]

Open sets from stop region that probably can be reached from the start region:
[0.5 1 1] ... [1 2 2]
[1 1 1] ... [2 2 2]
[2 1 1] ... [3 2 2]

Property IS NOT satisfied!
Maximum resolution reached!
   Number of iterations: 1
   Resolution reached: 0.5
   Approximate total run time: 1.19 secs
3.0 BIOLOGICAL MODELS

As a series of recognized biological functions or steps, biological processes represent basic or regulatory actions performed within a cellular system. Reactome is a knowledgebase of biological pathways involved in such processes for several species including humans [21]. In this paper, tryptophan regulation is the process of interest. Another process – gene transcription in *Vibrio fischeri* – with a simpler mathematical model is used to test the computational tools.

3.1 Tryptophan Regulation

Tryptophan is an essential amino acid, which means that it cannot be synthesized within the human body (or any vertebrates) and must therefore be ingested. Humans can obtain tryptophan from sources such as oats, bananas, milk, eggs, and fish, which contain tryptophan that has been produced by non-vertebrate organisms. The *Escherichia coli* (*E. coli*) bacterium is an example of an organism that can manufacture tryptophan. The chromosome* of *E. coli* contains five genes that encode for enzymes that produce tryptophan. The genes are preceded by a promoter: a sequence of the DNA strand which regulates expression of the genes. A gene regulatory protein can bind to a molecule called the operator to prevent or allow the promoter from being read. If the operator is free, another molecule called RNA polymerase can read the promoter and initiate transcription of the DNA strand in order to create the enzymes that manufacture tryptophan [1]. Figure 4 highlights the important steps of tryptophan biosynthesis.

* This word and all subsequent italicized words are defined in the Glossary.
When the level of tryptophan in the cell environment is high, tryptophan molecules bind to the tryptophan repressor (the gene regulatory protein for tryptophan regulation). This causes the repressor to become activated so that it can in turn bind to the operator. This prevents the promoter from being read, so that the genes cannot be transcribed and the corresponding enzymes cannot be created. Tryptophan production is therefore repressed.

When the level of tryptophan in the cell is low, the tryptophan repressor is no longer activated due to the lack of tryptophan molecules. The operator becomes free and the promoter can now be read by RNA polymerase so that the genes are transcribed [1]. The two states of tryptophan regulation can be represented by the hybrid system shown in Figure 5.
The biosynthesis of tryptophan is controlled by several elements in the cell environment. The metabolites involved in tryptophan regulation are shown in Table 3.

**Table 3: Metabolites Involved in Tryptophan Biosynthesis**

Taken From: [22]

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Metabolite</th>
</tr>
</thead>
<tbody>
<tr>
<td>chor</td>
<td>Chorismate</td>
</tr>
<tr>
<td>gln</td>
<td>Glutamine</td>
</tr>
<tr>
<td>glu</td>
<td>Glutamate</td>
</tr>
<tr>
<td>pyr</td>
<td>Pyruvate</td>
</tr>
<tr>
<td>an</td>
<td>Antranilate</td>
</tr>
<tr>
<td>prpp</td>
<td>Phosphoribosyl pyrophosphate</td>
</tr>
<tr>
<td>ppi</td>
<td>Pyrophosphate</td>
</tr>
<tr>
<td>npran</td>
<td>N-(5′-phosphoribosyl)-anthranilate</td>
</tr>
<tr>
<td>cpad5p</td>
<td>1-(O-Carboxyphenylamino)-1′-deoxyribulose-5′phosphate</td>
</tr>
<tr>
<td>co2</td>
<td>Carbon dioxide</td>
</tr>
<tr>
<td>igp</td>
<td>Indole glycerol phosphate</td>
</tr>
<tr>
<td>ser</td>
<td>Serine</td>
</tr>
<tr>
<td>t3p1</td>
<td>Glyceraldehyde 3-phosphate</td>
</tr>
<tr>
<td>trp</td>
<td>Tryptophan</td>
</tr>
</tbody>
</table>

These metabolites interact with each other in five chemical reactions that eventually result in the production of tryptophan. In the mathematical model for tryptophan
regulation, the state variables are the concentrations of the substrates that initiate the chemical reactions and the products that result. The five reactions are [22]:

\[
\begin{align*}
\text{chor} + \text{gln} & \rightarrow \text{glu} + \text{pyr} + an \\
\text{an} + \text{prpp} & \rightarrow \text{ppi} + \text{npran} \\
\text{npran} & \rightarrow \text{cpad5p} \\
\text{cpad5p} & \rightarrow \text{co2} + \text{igp} \\
\text{igp} + \text{ser} & \rightarrow \text{t3p1} + \text{trp}
\end{align*}
\]

In this paper, we focus on the last of these reactions, which involves \text{igp}, \text{ser}, \text{t3p1}, and \text{trp}. This chemical reaction is a basic enzymatic reaction of a form known as Michaelis-Menten. By applying quasi-steady state analysis, in which several enzymes involved in the reaction are assumed to disappear (go to zero concentration) after a short time period, the reaction can be simplified [23]. The resulting system of differential equations is:

\[
\begin{align*}
x_1 &= -k_1 x_1 x_3 \\
x_2 &= -k_2 c_o x_1 + k_2 c_o b_o - k_{-2} x_2 - k_3 x_2 \\
x_3 &= -k_1 x_1 x_3 + k_4 x_4 \\
x_4 &= k_3 x_2 - k_4 x_4 \\
x_5 &= k_3 x_2
\end{align*}
\]

where \(x_1\) is the concentration of \text{igp}, \(x_5\) is the concentration of \text{trp}, and the other state variables are concentrations of intermediate substrates and products in the Michaelis-Menten reactions. This system describes how the concentration of each metabolite changes with time. In this case, tryptophan production is turned on when the \text{trp} concentration becomes greater than approximately two [22]. This means that the desired target set is \(x_5 > 2\). The values of the parameters involved in the reactions are shown in the table below [22].

<table>
<thead>
<tr>
<th>Known Parameters</th>
<th>Unknown Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k_1 = 5.0)</td>
<td>(k_{-2})</td>
</tr>
<tr>
<td>(k_2 = 300)</td>
<td>(k_4)</td>
</tr>
<tr>
<td>(k_3 = 3.0)</td>
<td>(b_o) (initial concentration of \text{ser})</td>
</tr>
<tr>
<td>(c_o)</td>
<td></td>
</tr>
</tbody>
</table>
When all parameters are known, the five-dimensional model is as shown above. In order to solve the second project objective (determining the unknown parameters), we can modify the mathematical model by adding each unknown parameter as an additional state variable. The rate of change of a parameter over time is zero, since the parameter is a constant value. Presented below are the dynamics for variations of the tryptophan model which also include the unknown parameters. Unless the parameter is included as a state variable, the values for the each parameter were chosen to be $k_2 = 5$, $k_4 = 5$, $b_0 = 0.01$, and $c_o = 0.01$, based on typical values for reaction rates and concentrations.

**Dynamics when $k_2 (x_6)$ is unknown:**

$$x_1 = -k_1 x_1 x_3 = -5 x_1 x_3$$
$$x_2 = -k_2 c_o x_1 + k_2 c_o b_o - k_2 x_2 - k_2 x_2 = 0.03 - 3 x_1 - 3 x_2 - x_2 x_6$$
$$x_3 = -k_1 x_1 x_3 + k_4 x_4 = -5 x_1 x_3 + 5 x_4$$
$$x_4 = k_3 x_2 - k_4 x_4 = 3 x_2 - 5 x_4$$
$$x_5 = k_3 x_2 = 3 x_2$$
$$x_6 = 0$$

**Dynamics when $k_4 (x_6)$ is unknown:**

$$x_1 = -k_1 x_1 x_3 = -5 x_1 x_3$$
$$x_2 = -k_2 c_o x_1 + k_2 c_o b_o - k_2 x_2 - k_2 x_2 = 0.03 - 3 x_1 - 3 x_2 - 5 x_6$$
$$x_3 = -k_1 x_1 x_3 + k_4 x_4 = -5 x_1 x_3 + x_4 x_6$$
$$x_4 = k_3 x_2 - k_4 x_4 = 3 x_2 - x_4 x_6$$
$$x_5 = k_3 x_2 = 3 x_2$$
$$x_6 = 0$$
Dynamics when $k_2$ ($x_6$) and $k_d$ ($x_7$) are unknown:

\[
x_1 = -k_1x_1x_3 = -5x_1x_3
\]

\[
x_2 = -k_2c_0x_1 + k_2c_0b_o - k_{-2}x_2 - k_3x_2 = 0.03 - 3x_1 - 3x_2 - x_2x_6
\]

\[
x_3 = -k_1x_1x_3 + k_4x_4 = -5x_1x_3 + x_4x_7
\]

\[
x_4 = k_3x_2 - k_4x_4 = 3x_2 - x_4x_7
\]

\[
x_5 = k_1x_2 = 3x_2
\]

\[
x_6 = 0
\]

\[
x_7 = 0
\]

Dynamics when $k_2$ ($x_6$), $k_d$ ($x_7$), and $c_o$ ($x_8$) are unknown:

\[
x_1 = -k_1x_1x_3 = -5x_1x_3
\]

\[
x_2 = -k_2c_0x_1 + k_2c_0b_o - x_2x_6 - k_{-3}x_2 = -3x_2 - x_2x_6 + 3x_8 - 300x_1x_8
\]

\[
x_3 = -k_1x_1x_3 + x_4x_7 = -5x_1x_3 + x_4x_7
\]

\[
x_4 = k_3x_2 - x_4x_7 = 3x_2 - x_4x_7
\]

\[
x_5 = k_1x_2 = 3x_2
\]

\[
x_6 = 0
\]

\[
x_7 = 0
\]

\[
x_8 = 0
\]

Dynamics when $k_2$ ($x_6$), $k_d$ ($x_7$), $c_o$ ($x_8$), and $b_o$ ($x_9$) are unknown:

\[
x_1 = -k_1x_1x_3 = -5x_1x_3
\]

\[
x_2 = -k_2c_0x_1 + k_2c_0b_o - k_{-2}x_2 - k_3x_2 = -3x_2 - x_2x_6 - 300x_1x_9 + 300x_8x_9
\]

\[
x_3 = -k_1x_1x_3 + k_4x_4 = -5x_1x_3 + x_4x_7
\]

\[
x_4 = k_3x_2 - k_4x_4 = 3x_2 - x_4x_7
\]

\[
x_5 = k_1x_2 = 3x_2
\]

\[
x_6 = 0
\]

\[
x_7 = 0
\]

\[
x_8 = 0
\]

\[
x_9 = 0
\]
3.2 Gene Transcription in *Vibrio fischeri*

*Vibrio fischeri* (*V. fischeri*) is a marine bacterium whose luminescence can be controlled by the activation or deactivation of a set of *lux* genes. For example, *V. fischeri* is not luminescent as a free living organism. When the bacterium lives in symbiosis (in a mutually beneficial interaction) with a host marine organism, *V. fischeri* is typically luminescent [24]. Belta *et al.* simplified the traditional non-linear model of the luminescence of the bacterium to a set of three differential equations [24]:

\[
\begin{align*}
    x_1 &= k_2 x_2 - k_1 x_1 x_3 + u_1 \\
    x_2 &= k_1 x_1 x_3 - k_2 x_2 \\
    x_3 &= k_2 x_2 - k_1 x_1 x_3 - nx_3 + nu_2
\end{align*}
\]

where each state variable represents the concentration of different species involved in the *lux* gene activation. The desired behaviour of this system is such that when the state variables start in the set \(1 < x_i < 2\) for \(i = 1, 2, 3\), all trajectories are driven through the facet \(x_2 = 2\) in the positive \(x_2\) direction [24]. That is, the stop set (the complement of the target set) is \(x_2 < 2\) and the target set is \(x_2 > 2\) for initial conditions \(1 < x_i < 2\). This is an example of a system with control input \(u = [u_1, u_2]\). Belta *et al.* derived two sets of control inputs which lead the system to the target set. The two models are [24]:

**Model 1:**
\[
\begin{align*}
    u_1 &= -10(x_2 + x_1(x_3 - 1) - 4x_3) \\
    u_2 &= x_1(3 + x_2(x_3 - 1) - (x_2 - 2)x_3)
\end{align*}
\]

**Model 2:**
\[
\begin{align*}
    u_1 &= -10(x_2 + x_1(x_3 - 1) - 4x_3) \\
    u_2 &= 6
\end{align*}
\]

The values for \(u_1\) and \(u_2\) are plugged into the system of differential equations presented above. The two resulting dynamics will be used and referred to later in the paper. The set of dynamics when \(u_1 = u_2 = 0\) will be referred to as the uncontrolled vector field.
4.0 COMPUTATIONAL EXPERIMENTS

The first set of tests performed using RAMAS and the Level Set Toolbox were on the \textit{V. fischeri} model. The reasons for this were twofold: first, the \textit{V. fischeri} model has a smaller dimension that bounds the running time of the algorithms, and, second, for a certain stop set, the initial conditions are known from literature [24]. This allowed us to test the program with known inputs and outputs so that we could become familiar with the programs themselves as well as their limitations and typical output. This first set of tests were performed on both the Level Set Toolbox and RAMAS. The second set of tests performed were on the several variations of the tryptophan model and were used only with RAMAS. Since the Level Set Toolbox has been well documented and uses a well-known, recognized method to compute a solution, the focus was on obtaining the same degree of familiarity with RAMAS. The two test sets are presented below.

4.1 \textit{Vibrio fischeri} Test Sets

The initial test set focused on verifying the known initial conditions for the \textit{V. fischeri} model. The Level Set Toolbox and RAMAS were both used.

4.1.1 Level Set Toolbox Tests

As mentioned previously, the Level Set Toolbox accepts inputs for \textit{data0} and \textit{data}, which represent the target set and the initial set of conditions that reach the target. The Level Set Toolbox contains several functions that define basic shapes for the target set: cylinders,
rectangles, and planes. These shapes can be combined to create various other figures. Listed below are three different ways the target set \( x_2 > 2 \) could be described using the Level Set functions for a domain of \( 0 < x_l < 20 \):

**Set 1:** \( \text{data} = \text{shapeRectangleByCorners}(g, [0;2;0], [20;20;20]); \)  
(i.e. the rectangular shape of \( x_2 > 2 \))

**Set 2:** \( \text{data} = \text{shapeRectangleByCorners}(g, [0;2;0], [\text{Inf};\text{Inf};\text{Inf}]); \)  
(i.e. Set 1, but with infinity as the bounds of the shape instead of the maximum value of the domain)

**Set 3:** \( \text{data} = \text{shapeUnion}(\text{shapeRectangleByCorners}(g, [0;2;0], [20;20;20]), \text{shapeRectangleByCorners}(g, [1;1;1], [2;2;2])); \)  
(i.e. the initial set is also included in the target through a union – as opposed to an intersection – to the desired set \( x_2 > 2 \))

The following table shows the specifications for each set above and the total running time to reach a solution. The variables in the specification are as defined in Section 2.3. An additional input to the tool is the accuracy level, which can be low, medium, or high. In this case, we chose low accuracy for faster results. A test passed if the data set determined by the program included the initial set \( 1 < x_l < 2 \) known for the dynamics of the *V. fischeri* model.

**Table 5: V. fischeri Model on Level Set Toolbox**

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Pass?</th>
<th>Execution Time</th>
</tr>
</thead>
</table>
| A.1.    | Accuracy: Low  
tMax = 1.8    
Nx = 21       
g.min = [-5; -5; -5]  
g.max = [20; 20; 20]  
data0 = Set 1 | Yes          | 2136.59 seconds |
| A.2.    | Accuracy: Low  
tMax = 1.8    
Nx = 21       
g.min = [-5; -5; -5]  
g.max = [20; 20; 20]  
data0 = Set 2 | Yes          | 1814.77 seconds |
Since all three tests represent the same target set, we expect the reachable set computed to be the same in each case. Figure 6, Figure 7, and Figure 8 show the iterations performed by the tool to determine the initial conditions for Set 1, 2, and 3 respectively.

![Diagram](image)

Figure 6: *V. fischeri* Model on Level Set Toolbox – Initial Conditions for Target Set 1
Figure 7: *V. fischeri* Model on Level Set Toolbox – Initial Conditions for Target Set 2

Figure 8: *V. fischeri* Model on Level Set Toolbox – Initial Conditions for Target Set 3
In each case, we see that the set converges very quickly and does indeed include the initial conditions of $1 < x_i < 2$. To obtain more accurate results, the tests above could be re-run with a higher accuracy and a higher number of grid points. To keep the running time from increasing too much, $t_{Max}$ could also be reduced to less than 0.45 seconds, since the initial conditions set converges after three plots.

### 4.1.2 RAMAS Tests

Since RAMAS verifies a system by assuming the final set to be the complement of the target set, we need to encode $x_2 < 2$ as the undesired set (so that all trajectories are steered through the facet $x_2 = 2$ in the positive direction to the target set). Therefore, the stop set should be: $0 < x_2 < 2$ and $x_1$, $x_3$ any value in the domain. We choose the domain to start at zero since the state variables represent concentrations, which are non-negative.

Initially, the RAMAS tool was tested on a 2-dimension case study described in [17]. The same results were obtained as in [17] so the *V. fischeri* model was tested next. As mentioned in Section 3.0, there are three sets of dynamics available for the *V. fischeri* model (Model 1, Model 2, and the uncontrolled vector field). Table 6 shows various combinations of inputs to RAMAS. In each case, the start set is [1 1 1; 2 2 2]. All tests were performed on Models 1 and 2.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
</tr>
</thead>
</table>
| B.1.a.  | stop: $x_2 < 2$  
N = 10  
Domain = [0 N]  
Stop = [0 0 0; N 2 N] | No | No |
| B.1.b.  | N = 10  
Domain = [0 N]  
Stop = [0 0 0; 0 2 0] | No | No |
| B.2.a.  | stop: $x_2 < 2$  
N = 20  
Domain = [0 N]  
Stop = [0 0 0; N 2 N] | No | No |
In each test, 27 rectangles were generated and the output was produced in under four seconds. For Tests B.1.a., B.2.a., and B.3.a., we expect the final set to be unreachable since the controls were designed to lead the system to \( x_2 > 2 \) (and not the stop set \( x_2 < 2 \)). This is indeed the actual result. All other tests except Test B.5. encode \( x_1 \) and \( x_3 \) as singular points with \( 0 < x_2 < 2 \). Since \( x_2 \) is within the undesired range, we also expect the stop set to be unreachable. Test B.5. presents a contradicting result: when the stop set is the whole domain, we expect some states where \( x_2 > 2 \) to be reachable. Since this does not happen, a further series of tests was performed to investigate this discrepancy (Table 7). Again, the start set is [1 1 1; 2 2 2] and the tests were performed on Models 1 and 2.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.2.b.</td>
<td>( N = 20 )</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [0 0 0; 0 2 0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.3.a.</td>
<td>( N = 50 )</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [0 0 0; N 2 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.3.b.</td>
<td>( N = 50 )</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [0 0 0; 0 2 0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.4.</td>
<td>( N = 20 )</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [N 0 N; N 2 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B.5.</td>
<td>( N = 20 )</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = Domain</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7: \( V. fischeri \) Model on RAMAS – Second Test Set

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1.a.</td>
<td>( N = 100 )</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [0 2 0; N N N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.1.b.</td>
<td>( N = 1000 )</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Domain = [0 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [0 2 0; N N N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.2.</td>
<td>( N = 10 )</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Domain = [1 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = [2 2 2; N N N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.3.</td>
<td>( N = 10 )</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Domain = [1 N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop = Start = [1 1 1; 2 2 2]*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Test also performed for the uncontrolled vector field.
In this set of tests, the stop set was chosen to be $x_2 > 2$, which we expect to be reachable. Tests C.1. to C.2. show that the same discrepancy of Test B.5. appears. For Test C.3., we expect the stop to be reachable from the start since they are the same set. Again the actual and expected results are in conflict.

To resolve the discrepancies, the creators of the RAMAS software package were contacted. Upon discussing the results with them, the following important points and corrections were discovered:

- The domain vectors must include intermediate points that divide the range of each state variable. That is, the range of the continuous state space of the original system is partitioned into rectangles whose endpoints are defined by the values in the theta vector.
- While the program states that a maximum resolution of zero disables that input, inputting a zero implies that refinements of the original partition include new points that are very close to existing ones. At the next iteration of the algorithm, the number of points that need to be checked would then increase the computational complexity of the calculation.
- The number of iterations can typically be small: around two or three.
- The RAMAS program available online considers all types of rectangles (including points, lines, and full-dimension rectangles that have a non-zero width and length). Though this is a more general approach and could yield a greater set of results that can also identify specific points instead of whole areas of space as reachable, a program that considers only full-dimension rectangles can handle larger state spaces. The program that only handles full-dimension rectangles was obtained from one of the authors, Marius Kloetzer, and used on the tryptophan model.

Bearing the above observations in mind, the tests were re-run with revised inputs (Table 8). The domain in each case is [0 1 2 20], which includes all points that appear in the stop and start sets. The start set is [1 1 1; 2 2 2] and all tests were run on Models 1 and 2.
We expect Tests D.2., D.3., and D.4. to yield reachable sets since in each case $x_2 > 2$ is included in the stop set. Test D.3. is a specific case of Test D.2. where all three variables are restricted to be greater than two and Test D.4. is a generalization of Test D.2. where the stop is the whole domain. The results are correct in each of these cases. Test D.1. is the test which yields the result we are interested in: is $x_2 < 2$ reachable by some trajectory? We expect the answer to be no since the control inputs were derived to ensure that all trajectories driven through the facet $x_2 = 2$ in the positive direction. However, there are some reachable sets with $x_2 < 2$, which means that not all trajectories are driven through the facet in the desired direction. The reason for this discrepancy may be that the control inputs derived in [24] must satisfy the following constraints:

\[ 20 \leq u_i \leq 60 \]
\[ 1 \leq u_2 \leq 10 \]

With the chosen domain, we can find at least one set of points that does not satisfy the control inputs constraint: $x_1 = 10, x_2 = 2, x_3 = 3$ yields $u_1 = -100$ and $u_2 = 70$ for the first model ($u_2 = 6$ for the second model, but $u_1$ is still not within the appropriate range).
Because there are too many unknown variables, a smaller domain search was not attempted, but it may be possible to guess and test a range that could yield the correct results for Test D.1.

One of the input parameters to RAMAS is the resolution. To determine an appropriate resolution, Test D.1. was performed with resolution 0, 1, 2, 10, and 20. For a resolution of zero, an error occurred in the program and the reachable sets were not computed. This suggests that a refinement of the original partition could not be computed properly. For the other resolution values, the program ran without any error and the same results were computed. We can also see that in each test in Table 8, the resolution reached ($rp$) was one and the number of iterations performed was only one. From these results, we choose to use a resolution of six and a maximum of three iterations for the tryptophan model.

4.2 **Tryptophan Test Sets**

To solve the two objectives outlined in the Introduction, the information obtained from the initial testing of RAMAS was used to run the program on variations of the tryptophan model under different domain values, start sets, and stop sets. This section includes a series of tables that summarizes the important tests that were run.

In order to solve the initial conditions problem when all parameters of a model are known, we need to choose reasonable values for the unknown parameters and determine a set of initial conditions which do not lead to the undesired stop set. Since RAMAS is a model checking tool, we would ideally need to create a program that loops through all possible initial sets (and perhaps cleverly discards sets that should not work) in the domain. Due to timing constraints, this exhaustive search was not attempted in this project. Rather, several parameters and initial sets were tried in a wide range of values. For example, $k_d$ and $k_{2}$ values were tried with 0.01, 0.1, 5, and 100 in various combinations. For $b_o$ and $c_o$ the values 0.01 and 10 were tried. For the initial set, values
close to the threshold value \(x_5 = 2\) was chosen. The reason for this was to reduce the length that a trajectory would need to traverse and therefore make it more likely to reach a point beyond \(x_5 = 2\). However, a set of initial conditions were not found through this guess and test technique. Instead, we will focus on a set of values which does not satisfy the verification property (some of the initial values lead to the values outside the target set). This set was chosen after a brief review of related literature to determine reasonable values for the rates of reaction. The start set was chosen to match the small initial conditions values for \(b_0\) and \(c_0\). The specifications of this set of initial conditions is shown in Table 9.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1.</td>
<td>All parameters are known</td>
<td>Yes</td>
<td>X = 32 rectangles &lt;br&gt; t1 = 0.03 secs &lt;br&gt; t2 = 3.325 secs &lt;br&gt; t3 = 0.03 secs &lt;br&gt; rp = 1 &lt;br&gt; ip = 0.1 &lt;br&gt; t4 = 3.29474 secs</td>
</tr>
<tr>
<td></td>
<td>(k_4 = 5) &lt;br&gt; (c_0 = 0.01) &lt;br&gt; (b_0 = 0.01) &lt;br&gt; (k_2 = 5) &lt;br&gt; Domain = [0 0.1 0.2 2 10] &lt;br&gt; Start = [0.1 0.1 0.1 0.1 0.1; 0.2 0.2 0.2 0.2 0.2] &lt;br&gt; Stop = [0 0 0 0 0; 10 10 10 10 2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(r = 10) &lt;br&gt; (i = 2) &lt;br&gt; (d = 12)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In order to determine both the parameters and the initial conditions that lead to a certain target set, the models described in Section 3.0 were also input into RAMAS. This solves the second objective described in the Introduction. Initially, the start sets for these models were chosen at random in order to show the timing characteristics and ability of the program to converge to a solution for higher-dimension models. The set of results from this run are summarized in Table 10. In each case, \(r = 6\), \(i = 3\), and \(d = 12\).
Table 10: Higher Dimension Tryptophan Model on RAMAS

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.1.a.</td>
<td>( k_4 = 5 )  ( b_o = 0.01 )  ( c_o = 0.01 )  ( k_2 = x_6 ) (unknown parameter)  Domain = [0 0.1 0.2 4]  Start = [0.1 0.1 0.1 0.1 0.1 0.1; 0.2 0.2 0.2 0.2 0.2 0.2]  Stop = [0 0 0 0 0 0; 4 4 4 4 2 4]</td>
<td>Yes</td>
<td>N/A</td>
<td>X = 729 rectangles  ( t_1 = 0.282138 ) secs  ( t_2 = 270.794 ) secs  ( t_3 = 0.13508 ) secs  ( r_p = 1 )  ( t_4 = 271.109 ) secs</td>
</tr>
<tr>
<td>F.1.b.</td>
<td>Same as F.1.a. with start = almost all of domain</td>
<td>( k_4 = 5 )  ( b_o = 0.01 )  ( c_o = 0.01 )  ( k_2 = x_6 ) (unknown parameter)  Domain = [0 0.01 2 3.99 4]  Start = [0.01 0.01 0.01 0.01 0.01 0.01; 3.99 3.99 3.99 3.99 3.99 3.99]  Stop = [0 0 0 0 0 0; 4 4 4 4 2 4]</td>
<td>Yes</td>
<td>Yes (based on Test E.1. since this test includes the point ( k_2 = 5 ) and the initial conditions 0.1 to 0.2)</td>
</tr>
<tr>
<td>F.1.c.</td>
<td>Same as F.1.b. with stop = domain</td>
<td>( k_4 = 5 )  ( b_o = 0.01 )  ( c_o = 0.01 )  ( k_2 = x_6 ) (unknown parameter)  Domain = [0 0.01 3.99 4]  Start = [0.01 0.01 0.01 0.01 0.01 0.01; 3.99 3.99 3.99 3.99 3.99 3.99]  Stop = [0 0 0 0 0 0; 4 4 4 4 4 4]</td>
<td>Yes</td>
<td>Yes (based on Test E.1. since this test includes the point ( k_2 = 5 ) and the initial conditions 0.1 to 0.2)</td>
</tr>
<tr>
<td>F.1.d.</td>
<td>( k_4 ) is unknown</td>
<td>( k_2 = 5 )  ( b_o = 0.01 )  ( c_o = 0.01 )  ( k_4 = x_6 ) (unknown parameter)  Domain = [0 0.1 0.2 2 4]  Start = [0.1 0.1 0.1 0.1 0.1 0.1; 0.2 0.2 0.2 0.2 0.2 0.2]  Stop = [0 0 0 0 0 0; 4 4 4 4 2 4]</td>
<td>Yes</td>
<td>Yes (based on Test E.1. since this test includes the point ( k_4 = 5 ) and the initial conditions 0.1 to 0.2)</td>
</tr>
</tbody>
</table>
Tests similar to F.2. were also run for the 8-dimensional and 9-dimensional model, but the running time of the algorithm was over several hours and the program had to be stopped before a solution could be found.

Next, the higher-dimensional models were again run in order confirm that the results of the 5-dimensional model (Table 9) and the higher-dimensional model agree. These results are summarized in Table 11 (6-dimensional model) and Table 12 (7-dimensional model). In each case, $r = 6$, $i = 3$, and $d = 12$. The expected results are described in each table.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.1.a.</td>
<td>$k_2$ is unknown (initially between 4.5 and 5.5 which includes the value of 5 assumed in Test E.1.)</td>
<td>No</td>
<td>N/A</td>
<td>$X = 4096$ rectangles t1 = 0.69836 secs t2 = 608.585 secs t3 = 0.149426 secs ip = 1 rp = 0.5 t4 = 597.038 secs</td>
</tr>
<tr>
<td>G.1.b.</td>
<td>Dimension 6 (k_2) is unknown</td>
<td>Test G.1.a. with stop set up to 6 instead of 4.</td>
<td>Domain = [0 2 4.5 5.5 6] Start = [0 0 0 0 0 4.5; 6 6 6 6 6 5.5] Stop = [0 0 0 0 0 0; 6 6 6 6 2 6])</td>
<td>Yes</td>
</tr>
<tr>
<td>G.1.c.</td>
<td>Dimension 6 (k_2) is unknown</td>
<td>(k_2) is initially between 4 and 6, rest of variables are whole domain</td>
<td>Domain = [0 2 4.0 6] Start = [0 0 0 0 4.0; 6 6 6 6 6] Stop = [0 0 0 0 0; 6 6 6 6 2]</td>
<td>Yes</td>
</tr>
<tr>
<td>G.2.a.</td>
<td>Dimension 6 (k_2) is unknown</td>
<td>Start set is as in Test E.1. and (k_2) is between 4 and 6, which includes the value of 5 assumed in Test E.1.</td>
<td>Domain = [0 0.1 0.2 2 4.0 6] Start = [0.1 0.1 0.1 0.1 4.0; 0.2 0.2 0.2 0.2 0.2 6] Stop = [0 0 0 0 0 0; 6 6 6 6 2 6]</td>
<td>Out of memory error.</td>
</tr>
<tr>
<td>G.2.b.</td>
<td>Dimension 6 (k_2) is unknown</td>
<td>Test G.2.a. with all variables except (k_2) between 0 and 2.</td>
<td>Domain = [0 2 4.0 6] Start = [0 0 0 0 4.0; 2 2 2 2 6] Stop = [0 0 0 0 0; 6 6 6 6 2 6]</td>
<td>Yes</td>
</tr>
</tbody>
</table>
G.2.c.  
Dimension 6  
k_2 is unknown  
Test G.2.a. with  
k_2 on a smaller range  
Domain = [0.1 0.2 2 4.8 5.2]  
Start = [0.1 0.1 0.1 0.1 0.1 4.8; 0.2 0.2 0.2 0.2 0.2 5.2]  
Stop = [0.1 0.1 0.1 0.1 0.1 0.1; 5.2 5.2 5.2 5.2 5.2 5.2]  
Yes  
Yes  
(based on Test E.1.)  
X = 4096 rectangles  
t1 = 1.63278 secs  
t2 = 1685.65 secs  
t3 = 18.1761 secs  
ip = 1  
rp = 0.1  
t4 = 1687.77 secs

G.2.d.  
Dimension 6  
k_2 is unknown  
Test G.2.a. with  
k_2 on a smaller range  
Domain = [0.1 0.2 2 4.9 5.1]  
Start = [0.1 0.1 0.1 0.1 0.1 4.9; 0.2 0.2 0.2 0.2 0.2 5.1]  
Stop = [0.1 0.1 0.1 0.1 0.1 0.1; 5.1 5.1 5.1 5.1 5.1 5.1]  
Yes  
Yes  
(based on Test E.1.)  
X = 4096 rectangles  
t1 = 1.73599 secs  
t2 = 1669.64 secs  
t3 = 4.9373 secs  
ip = 1  
rp = 0.1  
t4 = 1670.13 secs

Table 12: Dimension 7 Tryptophan Model on RAMAS

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Specification</th>
<th>Stop Reachable? (actual result)</th>
<th>Stop Reachable? (expected)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.3.a.</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>
| Dimension 7  
k_2, k_4 are unknown  
Start as in Test E.1. and k_2, k_4 between 4.5 and 5.5, which includes the point 5 assumed in Test E.1.  
Domain = [0 0.1 0.2 4.5 5.5 6]  
Start = [0.1 0.1 0.1 0.1 0.1 4.5 4.5; 0.2 0.2 0.2 0.2 0.2 5.5 5.5]  
Stop = [0 0 0 0 0 0 0; 6 6 6 6 2 6 6]  
Out of memory error.  | Yes  | Yes  (based on Test E.1.)  | X = 78125 rectangles  
t1 = 16.2234 secs  
t2 = 48817 secs |
| G.3.b.  |               |                                 |                             |        |
| Dimension 7  
k_2, k_4 are unknown  
k_2, k_4 between 4.5 and 5.5, rest of variables are whole domain initially  
Domain = [0 2 4.5 5.5]  
Start = [0 0 0 0 0 4.5 4.5; 5.5 5.5 5.5 5.5 5.5 5.5 5.5]  
Stop = [0 0 0 0 0 0; 5.5 5.5 5.5 5.5 2 5.5 5.5]  | Yes  | Yes  (based on Test E.1. since this test includes the point k_2 = 5 and the initial conditions 0.1 to 0.2)  | X = 2187 rectangles  
t1 = 0.95578 secs  
t2 = 3443.52 secs  
t3 = 0.132115 secs  
ip = 1  
rp = 1  
t4 = 3432.17 secs |
In each case, the expected results agree with the actual results. For example, in the 5-dimensional model we assumed $k_2=5$ and the results yielded a system where the stop is reachable from the start. So it is expected that a higher-dimensional system which includes the point $k_2=5$ in the initial conditions and is otherwise operating under the same start/stop conditions will yield the same reachability result. If we started with values for the parameters in the 5-dimensional model which resulted in unreachable stop set (and a reachable target set), we would expect the results of higher-dimensional models to also give a system that does not fall outside the target. Tests G.2.c, G.2.d., G.3.c., and G.3.d. would be especially insightful in this case, since it would be possible to narrow down a value for the unknown parameter which would lead the system from reaching undesired states to reaching only the desired target. Test G.1.a. is also a useful test because it shows us that if the stop set is restricted to values from zero to four for variables $x_1$ to $x_5$, the undesired set is not reached. This may indicate that restricting the domain from zero to four would lead to the identification of a set of initial conditions that do lead us to the target.
Finally, we present a summary of the computational performance of the RAMAS tool.
From the various tests performed, the following factors seem to determine the running time of the algorithm:

- The number of values included in the domain. For example, in Test G.2.a. (with a dimension of six and six values in the domain) Matlab ran out of memory while some of the higher-dimension models (with only four values in the domain) completed within a reasonable time period.
- The dimension of the system.
- The number of rectangles that partition the state space. The trend is that for more rectangles, the running time is higher.

The following table averages the running times of various tests run on the RAMAS tool.

<table>
<thead>
<tr>
<th>Test IDs</th>
<th>No. Elements in Domain</th>
<th>Dimension (n)</th>
<th>No. Rectangles (X)</th>
<th>Total Running Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1.</td>
<td>5</td>
<td>5</td>
<td>32</td>
<td>3.29474</td>
</tr>
<tr>
<td>F.1.a., F.1.c., G.1.c., G.2.b.</td>
<td>4</td>
<td>6</td>
<td>729</td>
<td>290.67175</td>
</tr>
<tr>
<td>F.1.b., F.1.d., G.1.a., G.1.b., G.2.e., G.2.d.</td>
<td>5</td>
<td>6</td>
<td>4096</td>
<td>1486.25</td>
</tr>
<tr>
<td>G.2.a.</td>
<td>6</td>
<td>6</td>
<td>15625</td>
<td>Above 2351.96</td>
</tr>
<tr>
<td>F.2.a., G.3.b., G.3.c., G.3.d.</td>
<td>4</td>
<td>7</td>
<td>2187</td>
<td>3354.83</td>
</tr>
<tr>
<td>G.3.a.</td>
<td>6</td>
<td>7</td>
<td>78125</td>
<td>Above 48817</td>
</tr>
</tbody>
</table>

We can see that if the dimension is held constant and the number of elements in the domain is increased, the number of rectangles and the total running time also increases. Further, we see that the number of rectangles created by the program is dependant on both the dimension and the number of elements in the domain (so that four elements at dimension six yields less rectangles than four elements at dimension seven). We can also
see that if a dimension is fixed, running time depends only on the number of rectangles, but across dimensions running time also depends on \( n \) (so that 4096 rectangles at dimension six takes a shorter time to finish than only 2178 rectangles at dimension seven). Therefore, the running time depends on both the number of rectangles and the dimension \( n \). If the number of rectangles is less than 32, the program returns results within a few seconds. If the number of rectangles increased beyond 15625, it is likely that Matlab will run out of memory before a result can be computed.

We also briefly include an important note on the limitations of inputs to RAMAS. Since RAMAS handles multi-affine systems, there are \( 2^n \) combinations of state variables which could be terms in each differential equation \((x_1, x_2, x_1x_2, x_3, x_1x_3, \text{ etc})\). The RAMAS input for the dynamics (input Model) expects the coefficients of all of the \( 2^n \) terms to be entered, as described in the Appendix. For a system with dimension six, this is already 64 columns. Another limitation of fast testing was the user-interface of RAMAS. While this is generally a friendly aspect of a computational tool, running a large number of tests is made more efficient if the input is automated in another way. To resolve these two drawbacks, a C++ program was written to create a Matlab script file with the input values already defined. The C++ program reads from an input file the domain, start set, stop set, resolution, number of iterations, number of digits, and model (which requires only the non-zero coefficients of each term to be entered). Since there are very few non-zero terms in the tryptophan and \( V. \) fischeri model, this simplification eliminates writing out zeros for each possible term that could be formed from the state variables. The C++ program generates a Matlab script that can be run similar to the original RAMAS tool. Since the input values have already been entered, the program immediately begins computing a result and the user is not asked for any more inputs.
5.0 RESULTS AND RECOMMENDATIONS

Having explored both the Level Set Toolbox and RAMAS with models of biological processes, we can analyze the performance of each tool according to four metrics: generality, speed, accuracy, and ease of use.

In terms of generality, the Level Set Toolbox has the greater degree of generality, as it can provide an exact solution for any type of system. RAMAS is limited only to multi-affine system, but many biological processes are either of this form or can be simplified to this form.

In terms of speed, the Level Set Toolbox is extremely inefficient for systems with a dimension greater than five since its running time complexity is exponential. For the 3-dimensional \textit{V. fischeri} model, approximately 30 minutes were required to obtain the initial conditions at low accuracy. This would increase significantly for the 5-dimensional tryptophan model. The authors of RAMAS claim that the tool can handle systems up to a dimension of ten \cite{17}. If the number of elements in the domain are kept to a minimum, we believe it is possible that RAMAS will give a result within a few hours. For example, a 7-dimension system returned an output within an hour (with, of course, only four elements in the domain).

The Level Set Toolbox produces an exact solution to a given reachability problem. In this sense, as a direct method, it is the most accurate technique. RAMAS partitions the domain and conservatively creates trajectories in each partition it creates. Therefore, the reachable sets calculated by RAMAS are not exact and based only upon a small, finite set of values that divide each axis (the \textit{theta} vectors described in Section 2.3).
In terms of ease of use, the Level Set Toolbox does not provide an interactive user interface, so the user must search through the code and change variables they wish to modify. However, the program is well documented and maintained. RAMAS has less documentation and fewer examples. Because its user interface was inefficient for running a large set of similar tests, it was necessary to automate the input-output sequence by writing a program that ran RAMAS on parameters in an input file.

RAMAS provides several advantages to researchers and scientists looking for quick results on which they can base their initial estimates. For example, it would be efficient to check a hypothesis on RAMAS within a short time period, and, if the results are favourable, confirm or obtain a more accurate result from the longer-running Level Set Toolbox. In the case when the dimension of the system is high, RAMAS is also a valuable and necessary asset.

A set of initial conditions that lead the tryptophan model to a state of $x_2 > 2$ was not determined. Instead, we presented a general procedure for finding these sets and also a possible particular set which may result in an acceptable initial condition (see Test G.1.a.). By including the unknown parameters as additional state variables, a technique for solving the parameter problem and the initial conditions problem simultaneously was demonstrated. This technique, which also allows results of higher-dimension systems to be confirmed with lower-dimension systems, seems to have great potential especially when used with the RAMAS.

There are several directions in which this project can be taken. First, it was observed that in many cases, the resolution obtained by the tool was 0.1. It would be useful to explore whether or not reducing the maximum resolution ($r$) would change any of the results.

Second, regarding the Level Set Toolbox, the $V. fischeri$ model could be re-run at a higher accuracy with a higher number of grid points and a smaller finish time. In a typical experiment, the Level Set Toolbox tests presented in this paper would be considered a rough estimate used to determine appropriate values for the number of grid points and
finish time. To enhance the accuracy of the results, the test would then be run again as stated, with an expected total running time reaching a few hours. The Level Set Toolbox could also be run on the 5-dimensional tryptophan model to confirm the results of Test E.1. from RAMAS. Third, regarding the tryptophan model on RAMAS, a program could be written that systematically checks various initial condition values until a set is found that satisfies the safety property. This would require identifying an efficient order in which to check the initial conditions so that no viable point is overlooked and so that new points can be discarded based on previous results. For example, consider looking at the reachable sets computed for a particular set of initial conditions. If there are very few reachable sets that include $x_5 < 2$, then we know this set of initial conditions is very close to satisfying the reachability property (since only a few trajectories lead outside the target). We can take that particular set of initial conditions are try other values that are close. Lastly, if a set of initial conditions is found, the tool can be run on a higher-dimensional model with the same initial conditions and values of each parameter that are close to the 5-dimensional case. For example, if the unknown parameter was 2 in the 5-dimensional case, the initial conditions for the higher-dimensional case could be 1.5 to 2.5. If it is still that case that no trajectories lead outside the target, then we can conclude that the system is robust up to a 0.5 variation in that parameter. If that range is slowly increased, we could potentially determine a threshold where the parameter value has deviated far enough from its stable value that it makes the system reach an undesirable state.
6.0 CONCLUSION

This report investigated how casting a biological process into a mathematical framework is useful in simulating the system under various conditions. Hybrid systems can be used to characterize processes – like tryptophan regulation – that switch between a finite number of states obeying continuous dynamics. Two defining features of the mathematical model of tryptophan regulation are the constant, unknown parameters that appear in the differential equations and the initial conditions of the system. Given a target state that the system must lead to, the goal was to determine both the parameters and the initial conditions.

Reachability and model checking are two techniques which can solve the initial conditions problem. The two tools studied in this paper, the Level Set Toolbox and RAMAS, represent reachability and model checking, respectively. The Level Set Toolbox is a direct method and uses the Hamilton-Jacobi partial differential equation to evolve the boundary of the system backwards in time. The Level Set Toolbox produces an exact solution for all types of systems, but is very expensive computationally and inefficient for systems with dimension greater than five. RAMAS is an indirect method that partitions the continuous domain into discrete quotients representing the trajectories of the original system. Because RAMAS uses an approximation of the original system, the results produced are more conservative but faster to obtain. RAMAS is applicable only to multi-affine dynamics.

The Level Set Toolbox and RAMAS were first tested on a simple, 3-dimensional model of luminescence in the marine bacterium *V. fischeri*. Discrepancies produced by RAMAS were clarified upon discussion with the authors of the paper and useful information was obtained about typical inputs to the maximum resolution, the maximum number of iterations, and the number of digits used in the computation. The tool was then ready to
use on the tryptophan regulation model. The complete regulation model was simplified by considering only one of the reactions involved in tryptophan biosynthesis. The resulting 5-dimensional model with four unknown parameters led to the creation of a small program automated the testing procedure by reading inputs from a file in lieu of the user-interface of the original RAMAS program. To solve the parameter problem and initial conditions problem simultaneously, the unknown parameters were included as additional state variables in the dynamics of the system. A primary set of testing revealed no initial conditions that could reach the target set. A more comprehensive search must be performed to solve the first objective of this project by identifying these values using RAMAS. However, using a set of initial conditions that include trajectories which do not all reach the target set, results at a higher dimension confirmed those obtained from the original 5-dimensional model.

This project can be expanded in the following ways for tryptophan regulation:

1. Develop a program that successively tests initial points using RAMAS, in order to determine all possible initial conditions that lead to the desired target set of a tryptophan concentration greater than two.

2. Repeat the series of higher-dimension tests outlined in this paper with the initial conditions found in 1.), while varying the range of the unknown parameters in order to determine their robustness.

By developing an efficient method of combining the initial conditions problem and the parameter problem, a complete model – consisting of both the viable parameters and initial conditions – could be obtained more efficiently. Knowing what starting points lead to a target state can aid biologists in testing their hypotheses in a simulation environment without wasting expensive laboratory resources. The Level Set Toolbox and especially RAMAS have the potential to realize this goal. With further testing and exploration, we are confident that RAMAS could be used to obtain fast and insightful results for biological processes like tryptophan regulation.
REFERENCES


APPENDIX

The RAMAS tool accepts as an input parameter the dynamics of the system, labeled as Model in the input/output sequence shown in Section 2.4. Model is a matrix of size $n$ by $2^n$ which represents the system of differential equations that identify the dynamics of the system. Each column of the matrix holds the coefficient of one of the terms possible for an $n$-dimension system. For example, the input for the 3-dimension uncontrolled vector field of the V. fischeri system (given in Section 3.2) would look as follows:

**Table 14: Vector Field Coefficients for V. fischeri Uncontrolled Vector Field Model on RAMAS**

<table>
<thead>
<tr>
<th></th>
<th>constant term</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_1x_2$</th>
<th>$x_3$</th>
<th>$x_1x_3$</th>
<th>$x_2x_3$</th>
<th>$x_1x_2x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>$k_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$k_2$</td>
<td>0</td>
<td>0</td>
<td>$k_1$</td>
<td>0</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$-n$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In Matlab, this could be represented as:

```matlab
M = [0 0 10 0 0 -30 0 0; 0 0 -10 0 0 30 0 0; 0 0 10 0 -10 -30 0 0]
```

As the dimension of the system increases by one, the number of columns double. Since the tryptophan model has only a few non-zero coefficients, a more efficient Matlab representation could be used. For the V. fischeri model, it would look like:

```matlab
M = zeros(3,3);
M(1,3) = 10;
M(1,6) = -30;
M(2,3) = -10;
M(2,6) = 30;
M(3,3) = 10;
M(3,5) = -10;
M(3,6) = -30;
```
With this notation, only the non-zero values need to be entered. This format was used for the tryptophan model, which requires up to 512 columns for the models presented in this paper.

The terms in each column appear in a specific order, as determined from discussion with one of the authors of RAMAS, Marius Kloetzer. Consider trying to determine the column number at which the coefficient of term \(x_1x_3\) should appear. The steps that need to be followed are:

1. Create a binary number with \(n\) zeros:
   
   000 (since \(n = 3\) for \(V. fischeri\))

2. Number the position of each digit in the result step 1, where the first position is the right-most digit and \(n^{th}\) position is the left-most digit:
   
   pos. 3, pos. 2, pos. 1

3. If variable \(x_i\) appears in the term, place a 1 at position \(i\) in the binary number from step 1:
   
   001 for \(x_1\) (position 1) and 100 for \(x_3\) (position 3), resulting in 101

4. Convert the binary number from step 3 to a decimal number:
   
   \(101_2 = 5_{10}\)

5. Add 1 to the decimal value from step 4. This is the column number:
   
   \(5 + 1 = 6\)

The procedure above was used to determine the column numbers for the coefficients of the tryptophan regulation models given in Section 3.1.