Sand2006-1993p

SAND REPORT

SAND2005-xxxx
Unlimited Release
Printed March 2006

Xyce™ Parallel Electronic Simulator

Biological Pathway Modeling and Simulation

Richard L. Schiek,
Elebeoboa E. May

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

SAND2005-xxxx
Unlimited Release
Printed March 2006

Xyce™ Parallel Electronic Simulator

Biological Pathway Modeling and Simulation

Richard L. Schiek
Electrical and Microsystems Modeling

Elebeoba E. May
Computational Biology

Sandia National Laboratories
P.O. Box 5800
Mail Stop 0316
Albuquerque, NM 87185-0316

March 29, 2006

Abstract

At the cellular level, biological regulation networks can be modeled as electrical circuits where signals are produced, propogated and sensed. Using **Xyce** it is possible to simulate large control networks consisting of entire cells or cell cultures in order to understand the dynamics and stability of such systems. This document describes methods and results on adapting **Xyce** to biological problems.

Acknowledgments

The authors would like to thank Grant Heffelfinger for providing the initial inspiration for this work. Additionally we would like to thank Eric Keiter, Rob Hoekstra, Scott Hutchinson and Tom Russo for their help in using **Xyce** in this new area.

Trademarks

Xyce™ Electronic Simulator and Xyce™ trademarks of Sandia Corporation.

Contacts

Richard Schiek rlschie@sandia.gov
Elebeoba May eemay@sandia.gov
World Wide Web http://www.cs.sandia.gov/xyce

Contents

1. Introduction **11**

2. Casting Biological Networks as Circuits **15**
2.1 Reactions and Conservation of Mass 16
2.2 Enzymatic Reactions 18
2.3 Diffusion 18
2.4 Circuit Primitives 19

3. Reaction Networks **21**
3.1 Oscillatory Reaction Systems 21
3.2 Coupling Oscillatory Systems 22

4. Multicellular Differentiation **27**
4.1 Drosophila Differentiation 27
4.2 Model Implementation 27
4.3 Simulation Approach 28
4.4 Results 30

5. Simulating Tryptophan and Lactose Regulatory Networks **33**
5.1 Application to *Escherichia coli* Regulatory Pathways 33
Boolean Kinetics Model of Gene Regulation 39
5.2 Lactose Operon 41

6. Simulation of an Eighty-One Node Inferred Gene Network **45**
6.1 Boolean Logic Framework for Simulation 45
Mathematical Logic and Boolean Algebra 45
6.2 Modeling Gene Networks with Boolean Logic 47
Circuit Constructs for Large-Scale Gene Network Modeling 47
6.3 Modeling and Simulation of Inferred Genetic Networks 51
Simulation of Whole-Cell Gene Network 51

7. Simulating *Escherichia coli* K-12 **57**
7.1 The Whole-Cell Metabolic Model 57
Circuit Components for Whole-Cell Model 57
Constructing the $E.coli$ Metabolic Netlist 60

7.2 The Hybrid Whole-Cell Model 60
Coupling Metabolic and Genetic Components 63

8. Tryptophan Revisited: Reformulating the Metabolic Subcircuit 67
8.1 Reformulation of the Metabolic BSOURCE Element 67
8.2 Optimization Framework for Model Refinement 72

Appendix

A Chemcial channel 77
B Power law based reactions 79
C Enzymaticly controlled reactions 99

Figures

1.1 A simple genetic switch where the expression of genes is controlled by other gene products and environmental factors. 12
1.2 An electrical circuit analog of a simple genetic switch. 13

3.1 A simplified BZ reaction system confined to a cell. Each cell contains a cyclical reaction pathway which produces oscillating concentrations of species, B and C. Reaction rates are concentration dependent and influenced by neighboring cells through a diffusive connection to a common environment. 22
3.2 Ten BZ cells in a line producing concentration waves of species B. As species A diffuses out from the origin, the cells initiate diffusion-limited, concentration waves of species B. 23
3.3 Fifty BZ cells, simulating the entire BZ reaction system, in a line demonstrating signal transduction along the cellular axis. 25

4.1 A *Drosophila sp.* embryo showing developmental differentiation. [1] 28
4.2 A Developmental control circuit in *Drosophila sp.* 29
4.3 Celluar levels of differentiation promoters. 30
4.4 Resulting segment number histogram with noisy input. 31
4.5 Differentiation response to input noise. 32

5.1 Genetic and metabolic control for the activation of tryptophan biosynthesis. 34
5.2 Genetic and metabolic control for the repression of tryptophan biosynthesis. 35
5.3 Circuit diagram of tryptophan genetic and metabolic control 37
5.4 Circuit diagram of tryptophan genetic and metabolic control. 39
5.5 Simulation results of tryptophan genetic and metabolic circuit using a Boolean kinetics framework for gene network modeling. 40
5.6 Circuit diagram of lactose genetic and metabolic control 42
5.7 Circuit diagram of lactose genetic and metabolic control 44

6.1 Biological parallels to hardware system abstraction layers. 46
6.2 Truth table (A), logic gate schematic (B), and Boolean equation (C) representation of B-cell population (from Kaufman, Urbain, and Thomas (1985) B-cell/T-cell discrete interaction model [Perlesnon and Weisbuch, 1997]). . . 48
6.3 Logic gate schematic representation of helper T-cell population (from Kaufman, Urbain, and Thomas (1985) B-cell/T-cell discrete interaction model [Perleson and Weisbuch 1997]. 49

6.4 Comparison of the output of Xyce simulation (blue, top curve) to discretized time series microarray data (red, bottom curve) for meta-gene node zero. . . 55
6.5 Comparison of Xyce simulation results (blue, top graphs) for meta-gene Node 44 and Nodes 0, 15, and 70 (inputs to Node 44) to discretized microarray data (red, bottom graphs). 56

7.1 Circuit schematic for a bidirectional metabolic reaction. 58
7.2 XDamp screenshot for the Xyce simulation results of *E. coli* whole-cell, circuit simulation. 61
7.3 The Covert-Palsson model of $E.coli$ central metabolism. 62
7.4 Xyce simulation (30 minutes) of hybrid $E.coli$ netlist. (Top) External glucose consumption and ($Bottom$) Cell biomass production. 64
7.5 Xyce simulation (30 minutes) of hybrid $E.coli$ netlist. (Top) External acetate diffuses into cell and ($Bottom$) internal acetate increases. 65

8.1 Comparison of Xyce tryptophan biosynthesis to experimental data. 75

Tables

2.1 Equivalents between the chemical, biological domain and the electrical circuit modeling domain. 15

5.1 Abbreviations for metabolites involved in tryptophan biosyntesis. 36

6.1 Boolean kinetics equations for implementation of genetic logic gates. 50
6.2 Example reduction of meta-gene three using Espresso. 52

7.1 Abbreviations for metabolites involved in example reversible reaction. 58

8.1 Multi-substrate/multi-product reaction rate formulation. 69

This page is left intentionally blank

1. Introduction

Expression of a genetic code defines characteristics of a given organism. As an organism grows and adapts to its local environment specific elements of its genetic code are expressed while other elements are suppressed. Complex control mechanisms exist to regulate the expression of genes during the life of a cell. [2, 3]

To fully appreciate how a genetic repository or genome translates into a functioning cell, one must understand the control mechanisms of genetic expression. Genetic products of a given gene can promote or suppress the further production of that gene creating a simple feedback loop. [2, 3] Similarly, genetic products from other genes can regulate the production of a given gene creating complex feedback loops or expression cascades. Feedback loops and cascades are not limited to a single cell, but can span an entire cell culture or cellular generation influencing differentiation and development. [4]

As an abstraction to better understand genetic expression and control, genetic material and its associated control mechanisms can be viewed as a genetic switch. [5, 2, 6] For example, figure 1.1 demonstrates a simple genetic switch involving two genes. Gene A is controlled by a promoter that is normally no allowing the expression of Gene A into RNA and a protein. The protein product from Gene A interacts with a normally off promoter for Gene B activating that promoter. On activation, Gene B is expressed and its protein product formed. To complete a cycle, the protein product from Gene B must bind with some environmental factor and if such binding occurs then this product will deactivate the promoter for Gene A. This simple switch will results in oscillatory concentration levels of the gene products over time. Also, this switch can be modeled as an electrical circuit where a signal (the transcript from a section of DNA) is generated and altered as it interacts with other components during its propagation. This analogy is far from perfect as there are significant differences in the switching speed and signal to noise ratio of a genetic circuit versus an electric circuit. However, this analogy allows one to consider very complicated, dynamic control circuits while investigating expression stability and population dynamics. [4]

Genetic expression and control pathways can be successfully modeled as electrical circuits. Given the vast quantity of genomic data, very large and complex genetic circuits can be constructed. To tackle such problems, the massively-parallel, electronic circuit simulator, **Xyce** [7], has being adapted to address biological problems. Unique to this bio-circuit simulator is the ability to simulate not just one or a set of genetic circuits in a cell, but many cells and their internal circuits interacting through a common environment. Additionally, the circuit simulator **Xyce** can couple to the optimization and uncertainty analysis framework

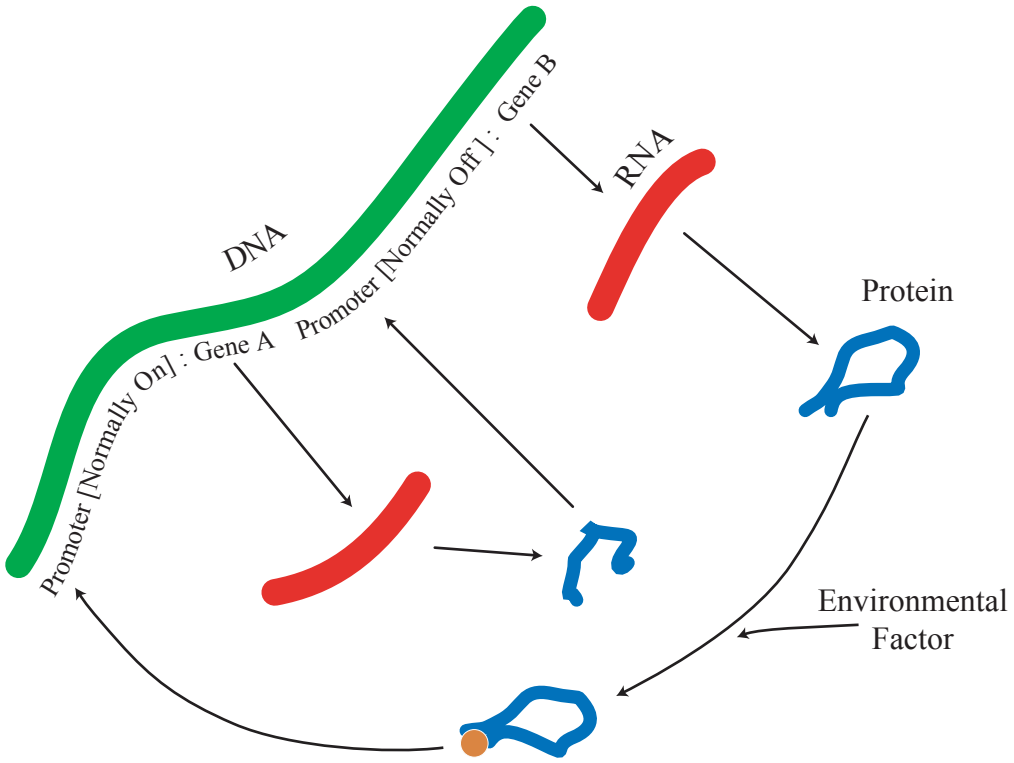

Figure 1.1. A simple genetic switch where the expression of genes is controlled by other gene products and environmental factors.

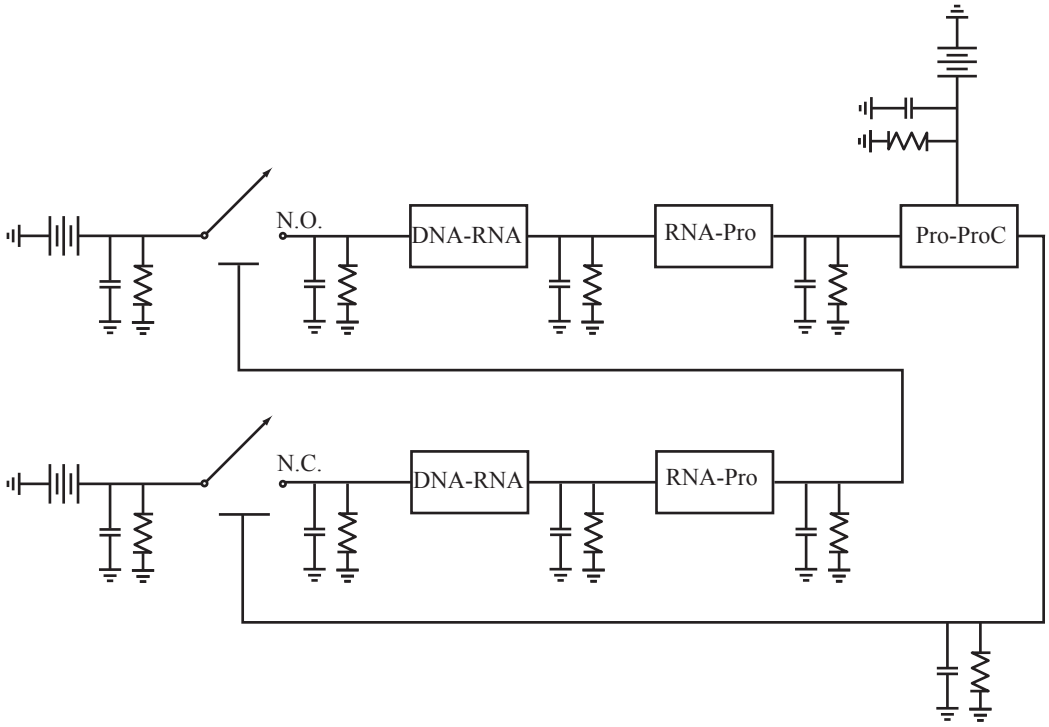

Figure 1.2. An electrical circuit analog of a simple genetic switch.

Dakota [8] allowing one to find viable parameter spaces for normal cell functionality and required parameter ranges for unknown or difficult to measure biological constants.

Circuit analogs for common biological and chemical machinery have been created. Using such analogs, one can construct expression, regulation and reaction networks. Individual species can be connected to other networks or cells via non-diffusive or diffusive channels (i.e. regions where species diffusion limits mass transport). Within any cell, a hierarchy of networks may exist operating at different time-scales to represent different aspects of cellular processes.

This simulator can model interesting biological and chemical systems such as the metabolic and genetic networks for an *E. coli* and *Drosophila sp.* cellular differentiation network. This document explores the construction of biological circuits, simulating partial and full cellular networks, connecting of many cells for cell culture simulations and visualizing the results.

2. Casting Biological Networks as Circuits

A biological or chemical simulation working within an electrical circuit context requires a translation framework to convert from the former domain to the latter. In electronics, a fundamental and conserved quantity is charge while in the biological domain, one is often concerned with concentration of a given chemical compound. Given a control volume, such as the volume of a cell, concentrations can be converted to mass. As a basis for a biological to electrical problem conversion, this work will equate mass of a given chemical species with charge. Each pathway or *wire* in a circuit will carry a different chemical species and the charge on that wire will denote the mass of that chemical species present in the simulation control volume.

Continuing this analogy, electrical current which is the timed rate of change of charge is equivalent to the rate of mass change, *i.e.* how quickly a compound is used or created by the system or mass flux through the system. Voltage is a relative measure of electrical potential. The chemical or biological analog to voltage here is chemical concentration provided one is measuring voltage relative to a neutral ground. Kirchhoff's Voltage Law, KVL, which requires the voltage drop around any closed circuit to be zero [9] enforces a stoichiometric balance on chemical reactions. Kirchhoff's Current Law, KCL, which requires the current flow into a circuit node balance current flow out of that node enforces conservation of mass within the system. For reference, table 2.1 summarizes the analogous terms joining chemical, biological problems and electronic circuit problems. It is important to note that with such a framework in place, we are not ignoring any important kinetic or stoichiometric aspects of the biological problem just to work within an electrical circuit domain. Rather, we are using this framework to take advantage of existing simulation capabilities

Chemical, Biological Domain	Electrical Domain
mass	charge
mass flux	current
concentration*	voltage
stoichiometric conservation	Kirchhoff's voltage law
mass conservation	Kirchhoff's current law

Table 2.1. Equivalents between the chemical, biological domain and the electrical circuit modeling domain.

developed for electrical modeling problems. Note, that there is no equivalent to volume in the electrical domain. Thus for one to strictly employ concentration one must first define a consistent system volume such as a biological cell or culture plate.

2.1 Reactions and Conservation of Mass

In a typical chemical or biochemical problem, one constructs a mass balance on the species of interest to understand dynamic behavior. Conservation of mass implies:

$$\text{Rate of mass change} = \text{Production} - \text{Consumption} + \text{Flux though boundary} \tag{2.1}$$

Specifically, for a given species, A, and a control volume V, equation 2.1 becomes:

$$\frac{dA}{dt} = \sum f_i(A) - \sum g_i(A) + \oint_{S(V)} h(A) \tag{2.2}$$

where the functions $f_i(A)$ represent the sources of A, $g_i(A)$ represents potential sinks of A, and the surface integral represents flux of A through the surface enclosing the control volume, $S(V)$.

To fully demonstrate how the conservation of mass inspired equation 2.2 translates into the electrical domain, consider the following reversible reaction system for compounds A, B and C:

$$aA + bB \rightleftharpoons cC \tag{2.3}$$

where a, b and c are stoichiometric coefficients. If k_f is the forward reaction rate constant and k_r is the reverse reaction rate constant and one assumes the reaction obeys power law kinetics, then rate of consumption of A by reaction 2.3 is:

$$\text{Consumption} = ak_f[A]^a[B]^b \tag{2.4}$$

and the rate of production of A from the reverse reaction of 2.3 is:

$$\text{Production} = ak_r[C]^c \tag{2.5}$$

Finally, if species A obeys fickian diffusion, any diffusive flux into or out of the control volume is:

$$\text{Flux} = k_d S(V)\left([A] - [A]_\infty\right) \tag{2.6}$$

Since diffusive flux is driven by a concentration gradient in A, a concentration of A outside of the local control volume V is required to fully specify a diffusive flux. This external concentration of is is denoted: A_∞, while the diffusion coefficient is k_d and the flux area is $S(V)$.

Note, that reaction rates are often calculated using species concentration. However, if one specifies a consistent control volume, *i.e.* a cell's volume or a finite element cell volume, then mass can be used in place of concentration. Since the concentration of A equals the mass of A divided by the control volume, $[A] = A/V$, one can use mass, A, rather than concentration, $[A]$, with a few extra multiplicative constants, VC_t,

Summing equations 2.4, 2.5 and 2.6 yields a conservation of mass equation for A.

$$\frac{d[A]}{dt} = ak_r[C]^c - ak_f[A]^a[B]^b + kS(V)\left([A] - [A]_\infty\right) \tag{2.7}$$

Rearranging terms in equation 2.7 and replacing concentration terms, $[\cdot]$, with voltage proposed in table 2.1 so that $[A]$ becomes V_A or voltage at node A everywhere except within the time derivative term.

$$\frac{d[A]}{dt} - ak_r V_C^c + ak_f V_A^a V_B^b - kS(V)\left(V_A - V_{A_\infty}\right) = 0 \tag{2.8}$$

Finally, within the time derivative we will convert the concentration of A to mass of A *via.* $[A] = A/V$. Also, following the proposed substitution of charge for mass from table 2.1, such that $A \rightarrow q_A$ equation 2.8 becomes:

$$\frac{1}{V}\frac{dq_A}{dt} - ak_r V_C^c + ak_f V_A^a V_B^b - kS(V)\left(V_A - V_{A_\infty}\right) = 0 \tag{2.9}$$

Equation 2.9 arose by applying a conservation of mass constraint on species A. Once translated to an electrical domain using the framework outlined in table 2.1, it now represents an application of Kirchhoff's current law to the circuit node, A. Examining the terms in equation 2.9 indicates the devices influencing node A. The first term represents a capacitor and has the physical meaning that the chemical A must be able to accumulate within the control volume. If one makes the logical choice of non-dimensionalizing the volume with the choice of control volume, then the capacitor capacitance will equal one implying that no work is required to accumulate species A. Terms two and three of equation 2.9

are voltage dependent current sources or *behavioral sources*. Current into our out of node A is dependent on the voltage at other nodes, just as the rate of production or consumption of species A depended on the concentration of other species. Finally, the last term represents a linear resistor adding or subtracting current depending on the voltage relative to a reference state such as ground. The exact form of the constant which would equal the resistors conductance depends on the problem geometry and is explored in the next section.

Physically this resistor serves several important purposes. First, in an electrical circuit, all voltages are relative quantities typically specified relative to a ground potential. Concentration on the other hand is not a relative quantity. Inserting a resistor between the node of interest and the ground ensures that the concentrations inferred from voltages are all relative to a ground or zero value. Second, this resistor provides a DC path to ground which is a requirement to consistently find a DC solution for this circuit. Finally, on a physical level, this resistor can represent common degradation process within a cell where a given chemical is continuously destroyed.

2.2 Enzymatic Reactions

While the previous example dealt only with power law kinetics, enzymatically controlled reactions mechanisms such as Michalis-Menton kinetics are easily handled. For example, given the simple, enzyme catalyzed reaction:

$$A \rightarrow B \tag{2.10}$$

applying the same conservation of mass approach as used above yields the following equation for circuit node A.

$$\frac{1}{V}\frac{dq_A}{dt} - V\text{max}\frac{V_A}{K_m + V_A} - kS(V)\left(V_A - V_{A_\infty}\right) = 0 \tag{2.11}$$

Note, that the overall structure of equation 2.11 is the same as equation 2.9 except that the behavioral source has changed to reflect the enzyme controlled kinetics.

2.3 Diffusion

The previous section dealt only with a reaction occurring at one point in space or a single node of a circuit. In a real system, reactions occur over a continuous span of distance and the reactants may not have uniform concentrations over that distance. Thus, one may need to consider how material is convected within the system in particular by diffusion.

The diffusion equation may be written as:

$$\frac{\partial A}{\partial t} - D\frac{\partial^2 A}{\partial x^2} = 0 \tag{2.12}$$

where A is the species of interest, t is time, x is a spatial coordinate and D is the diffusivity which must have units of length squared over time.

Since there is not a characteristic length scale in the circuit simulation the partial derivative in the spatial direction has no simple analog. Choosing a characteristic length scale from the chemical or biological domain denoted L, such as cell spacing, one can approximate this partial derivative with finite differences yielding

$$\frac{d[A]_i}{dt} - D\frac{[A]_{i+1} + [A]_{i-1}}{(2L)^2} = 0 \tag{2.13}$$

where the subscript i denotes a grid location.

Now, if one converts the concentration of A in the time derivative to a mass of A and to voltage as in previous section, the result is:

$$\frac{1}{V}\frac{dq_{Ai}}{dt} - \frac{D}{(2L)^2}\left(V_{Ai+1} + V_{Ai-1}\right) = 0 \tag{2.14}$$

One finds that using the diffusion equation to relate nodes in a circuit as if they were spatially connected by diffusion is equivalent to adding capacitors for species accumulation (fist term in equation 2.14) and linear resistors between the nodes (second term in equation 2.14). Importantly, one can relate the diffusivity and characteristic length scale to the conductance of the resistors: $D/(2L)^2$. Thus, to second order in the spatial dimension, one can fully capture the effects of diffusion. This simple analysis extends equally to two and three dimensions.

2.4 Circuit Primitives

With the above examples in mind, it is possible to generalize components of a reaction based network as devices in an circuit. These generalized devices can than be attached like regular circuit devices to create complex systems.

First, all species being tracked by a simulation must be able to accumulate and must have a simple path to ground. These qualification give rise to the *chemical channel* or *chemical wire* – a circuit connection where charge can accumulate, unlike a typical wire where no accumulation can take place. A simple subcircuit definition of a chemical channel is:

```
*
* Chemical channel subcircuit
*
```

```
* This is simple a capacitor to allow for the accumulation
* of material and a path to ground
*
.SUBCKT ChemCh chNode PARAMS: chCon=1e-16

Ccch1 chNode 0 1   IC=chCon
Rrch1 chNode 2 1e12

.ENDS
```

One could use such a subcircuit in a normal netlist with the line:

```
xWater    WaterNode  ChemCh PARAMS: chCon=55.5
```

Which for the chemical, Water, which exits on the node, WaterNode, would start the simulation with a concentration of 55.5 (the molar concentration of pure water).

3. Reaction Networks

In this chapter we consider networks of chemical reactions as a first step to modeling biological systems. Reaction networks occur in biological systems in the form of metabolic or biosynthesis pathways. Organic, chemical reactions are presented here as they are simpler and better understood than their biological counterparts. Additionally, when chemical networks are coupled together, self-organizing behaviors can be found.

3.1 Oscillatory Reaction Systems

The Belousov-Zhabotinskii (BZ) reaction system is a complex, chemical reaction network that produces oscillatory concentration waves of the metallic species Ce^{4+} and Ce^{3+}. [10] While interesting in its own, this reaction system is also a good analog of a biological reaction cycle.

$$
\begin{aligned}
HOBr + Br^- + H^+ &\rightleftharpoons Br_2 + H_2O && (3.1)\\
HBrO_2 + Br^- + H^+ &\rightleftharpoons 2HOBr && (3.2)\\
BrO_3 + Br^- + 2H^+ &\rightleftharpoons HBrO_2 + HOBr && (3.3)\\
2HBrO_2 &\rightleftharpoons BrO_3^- + HOBr + H^+ && (3.4)\\
BrO_3^- + HBrO_2 + H^+ &\rightleftharpoons 2BrO_2\cdot + H_2O && (3.5)\\
BrO_2\cdot + Ce^{3+} + H^+ &\rightleftharpoons HBrO_2 + Ce^{4+} && (3.6)\\
BrO_2\cdot + Ce^{4+} + H_2O &\rightleftharpoons BrO_3^- + Ce^{3+} + 2H^+ && (3.7)\\
Br_2 + CH_2(COOH)_2 &\rightarrow BrCH(COOH)_2 + Br^- + H^+ && (3.8)\\
6Ce^{4+} + CH_2(COOH)_2 + 2H_2O &\rightarrow 6Ce^{3+} + HCOOH + 2CO_2 + 6H^+ && (3.9)\\
4Ce^{4+} + BrCH(COOH)_2 + 2H_2O &\rightarrow 4Ce^{3+} + HCOOH + && (3.10)\\
&\quad Br^- + 2CO_2 + 5H^+ &&\\
Br_2 + HCOOH &\rightarrow 2Br^- + CO_2 + 2H^+ && (3.11)
\end{aligned}
$$

To simplify the modeling presented here, the key elements of the Belousov-Zhabotinskii reaction network can be summarized by the following system:

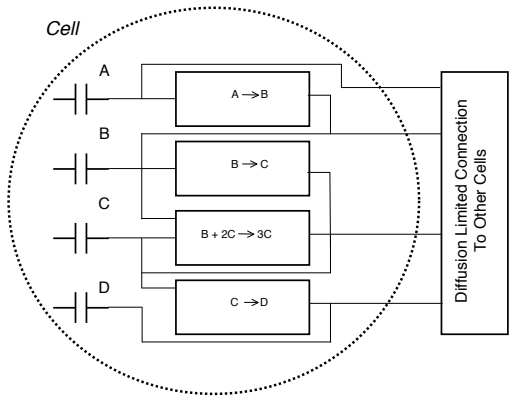

Figure 3.1. A simplified BZ reaction system confined to a cell. Each cell contains a cyclical reaction pathway which produces oscillating concentrations of species, B and C. Reaction rates are concentration dependent and influenced by neighboring cells through a diffusive connection to a common environment.

$$A \rightarrow B \tag{3.12}$$
$$B \rightarrow C \tag{3.13}$$
$$B + 2C \rightarrow 3C \tag{3.14}$$
$$C \rightarrow D \tag{3.15}$$

where the key to oscillatory behavior is the auto-catalytic production C in the third reaction where B combines with $2C$ to produce $3C$. This is the equivalent to the behavior of the radical species $BrO_2\cdot$ when one considers the sum of reactions 3.5 and 3.6 or 3.5 and 3.7 from the full Belousov-Zhabotinskii system.

3.2 Coupling Oscillatory Systems

Figure 3.1 depicts a simplified BZ reaction system known as the Oregonator [10] contained within a model cell, hereafter called a BZ reaction cell. For simplicity, the various chemical species have been renamed A, B, C and D. In the reaction scheme, species A is converted to B. Species B can be converted directly into C or, auto-catalytically using existing material C. Dual routes to the production of species C and different reaction rates lead to oscillations in the local concentrations of B and C as one reaction pathway is favored over the other. The last step in this reaction pathway is the conversion of C to species D.

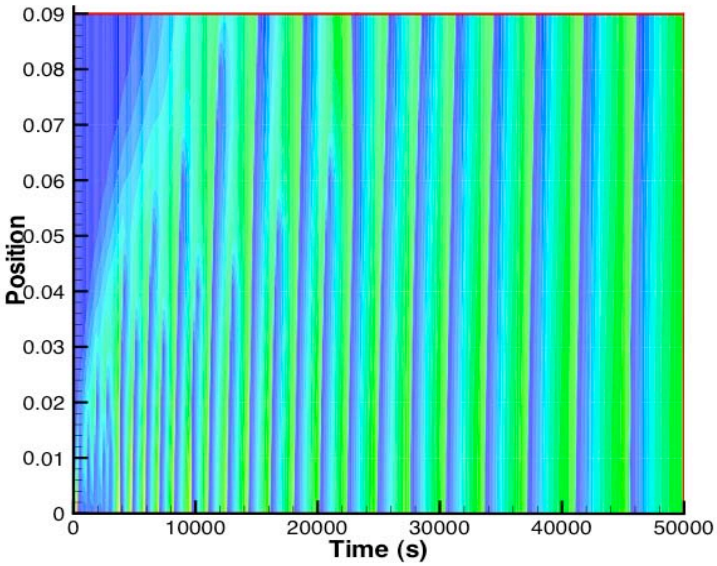

Figure 3.2. Ten BZ cells in a line producing concentration waves of species B. As species A diffuses out from the origin, the cells initiate diffusion-limited, concentration waves of species B.

Capacitors within the cell allow for accumulation mass for a given compound allowing concentrations to vary continuously. Each species is also free to diffuse into or out of the cell via a diffusive circuit element and such flux is proportional to the local concentration gradient. Finally, each chemical species can also participate in various chemical reactions as shown by the circuit lines connecting the various reaction devices.

Ten BZ reaction cells, arranged in a line and connected by a diffusion limited environment were simulated and the results are shown in figure 3.2. At time-zero, a drop of species A was placed at the origin. As time progresses, species A diffuses along the y-axis and also starts the reaction cycle in each cell. Since the reaction cycle produces time periodic concentrations of species B and C, one expects spatial variations in the concentration of species B. The contours shown in figure 3.2 depict the concentration of B, and they show diffusion-limited, concentration waves propagating along the line of cells. The concentration waves are due both to the diffusion of species A as well as diffusion of species B as it is consumed more quickly in neighboring cells that have locally higher concentrations of species A.

Returning to the full set of reactions listed at the start of this chapter, one can model signal transduction along a line of cells. Figure 3.3 presents a set of cells containing the entire Belousov-Zhabotinskii reaction system described earlier. As this system is considerably larger, the goal will be to show non-steady oscillation which here represents a signal being sent along the line of cells rather than a steady oscillation pattern. In figure 3.3, the cells are aligned along the *y*-axis and an initial pulse of Ce^3 at the origin. As time advances, a *signal* propagates along the cells in the form of a concentration wave of $BrO_2\cdot$ (or any of the oscillatory species). At longer times one starts to see the formation of a steady state behavior of pulses moving up the vertical axis.

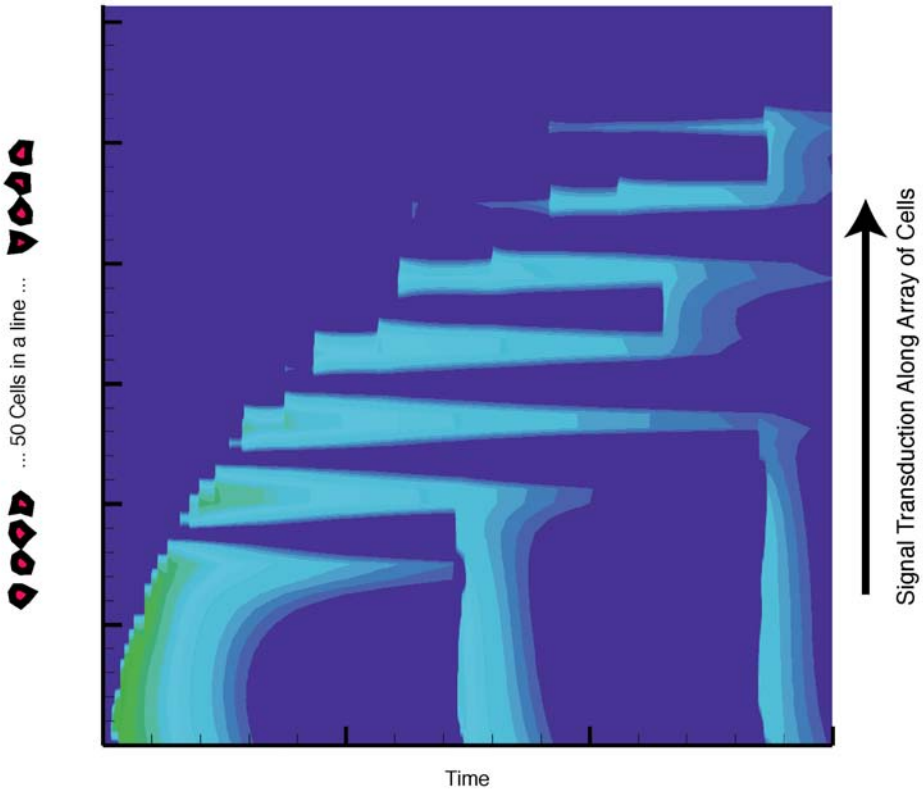

Figure 3.3. Fifty BZ cells, simulating the entire BZ reaction system, in a line demonstrating signal transduction along the cellular axis.

This page is left intentionally blank

4. Multicellular Differentiation

When biological networks interact across cell boundaries, communication is effectively occurring between these cells. In this chapter we examine how inter-cellular communication between networks controlling cellular development can lead to cells differentiating into various cell types. A well studied biological system is the fruit fly, *Drosophila sp.* to which we turn next.

4.1 Drosophila Differentiation

Development of a biological system from one state to another in a controlled manner usually involves feedback to assert such control. The multi-cellular network controlling tissue differentiation in the common fruit fly, *Drosophila sp.*, is no exception. [11, 12, 13] During *Drosophila's* development a series of bands develop along the major axis of the growing embryo (see micrograph in figure 4.1). Such bands are a graphical indicator of the underlying cellular differentiation in progress. The schematic shown in figure 4.2, represents the control network responsible for cellular differentiation in Drosophila. [11] Though complex, this network typically bifurcates into one of three states. If a cell is producing the gene product `wg` then the protein `WG` will likely be produced as well. The `WG` protein is exported into the cellular environment and picked up by neighboring cells where it can promote the expression of the gene product `en`. The `en` gene product represses the production of `wg` and puts the cell into a different state from a cell producing `WG`, specifically into a state where it is producing and expressing `HH`. Thus, cells will typically be producing either `WG` or `HH` with a small percentage of cells producing low levels of both of these proteins.

4.2 Model Implementation

To understand and model cellular differentiation, we have simulated the *Drosophila sp.* segment polarity gene network for a 2D array of cells connected through a common diffusion limited environment. In such an environment, cells experience local concentrations of differentiation stimuli determined by neighboring cells production and consumption rates. These local stimuli effect the genetic and metabolic regulatory networks within the cell directing the cells eventual development. For this model problem, we have examined func-

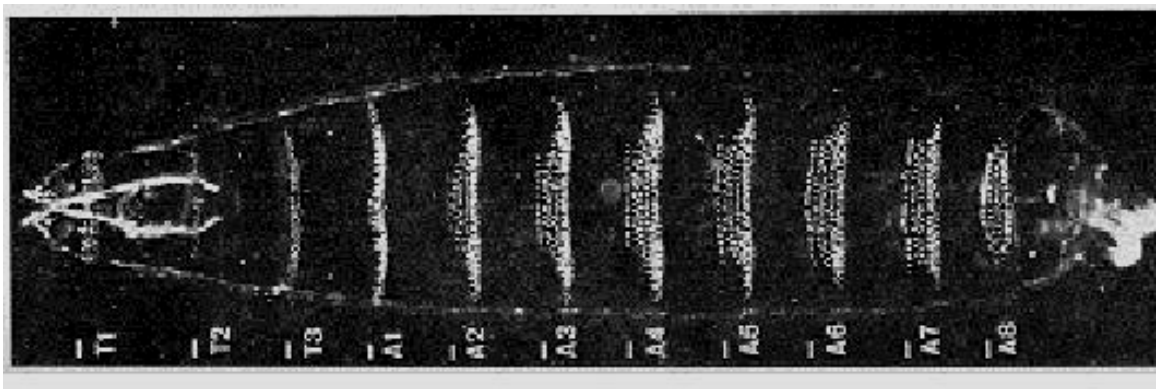

Figure 4.1. A *Drosophila sp.* embryo showing developmental differentiation. [1]

tionality and the system?s sensitivity to initial noise by using Dakota [8], Sandia's optimization program, to explore the system?s parameter space.

Actual simulations of the *Drosophila* network were carried out as follows. The network was converted to an electrical circuit using analogs for chemical reactions, material storage, promotion, repression, degradation and diffusion. These analogs treat electrical charge, a conserved quantity in electrical circuit simulators as mass. Once the intra-cellular circuit was created, a 10 by 10 grid of cells embedded within a diffusion limited environment was created, again as a circuit. Fundamental constants like reaction rates, enzymatic turnover rates and diffusion coefficients were parameterized within this circuit. Such parameterization allows the optimization program Dakota to alter parameters between simulation runs to explore the phase space for this system.

4.3 Simulation Approach

With expression, enzymatic turn-over, reaction and diffusion rates and noise levels all parameterized, a design of experiments approach with latin-hypercube sampling was used to understand how this collection of state variables controls the resulting system. To address the 22 primary model parameters, over 50,000 simulation runs were coordinated by Dakota and conducted by Xyce using a multi-level parallel computation approach. A statistical analysis of the simulation output allows one to gauge dominate control parameters and system stability relative to initial condition noise.

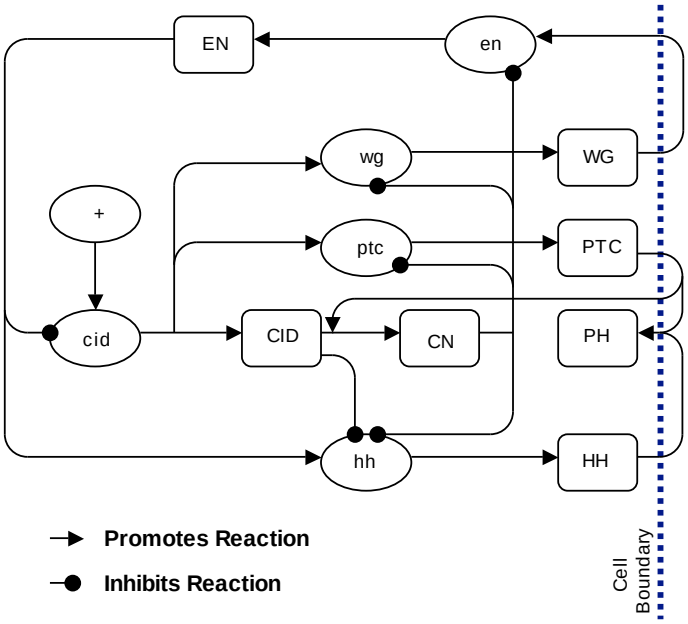

Figure 4.2. A Developmental control circuit in *Drosophila sp.*

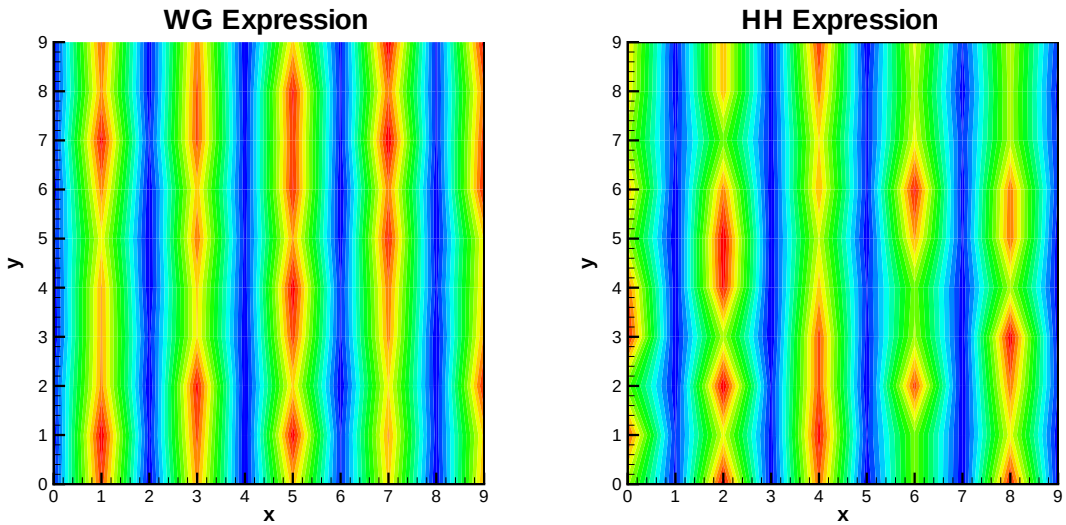

Figure 4.3. Celluar levels of differentiation promoters.

4.4 Results

Shown below in figure4.3 are concentration contour plots of the species `WG` and `HH`. Initially, the system was started with zero concentration of the exported species, `PH`, `PTC` and `HH` and an oscillatory level of `WG`. This initial oscillatory state represents the initial bias that anterior-posterior, dorso-ventral patterning hierarchies initiate in the developing embryo [12]. Additionally, a 10% rms. random noise was added to the `WG` initial conditions to simulate disturbances of the system from an ideal starting state. Such noise was also parameterized in the circuit and varied to gauge system robustness. Physically, the striations in concentration shown in figure 4.3 represent layers of cells becoming `WG` producing or `HH` producing over time similar to the micrograph of a Drosophila embryo shown in figure 4.1.

To study the effect of initial noise on the system's ability to differentiate, simulations were started with varying amounts of random noise in the `WG` concentration field. This noise was gaussian in distribution and ranged from 0 to 30% of the maximal value of `WG`. Figures 4.4 and 4.5 shows the probability of successful cellular differentiation as a function of initial system noise. While this system is very stable in the absence of noise [11, 13], this work demonstrates that even a small quantity of initial noise significantly reduces the system's functionality.

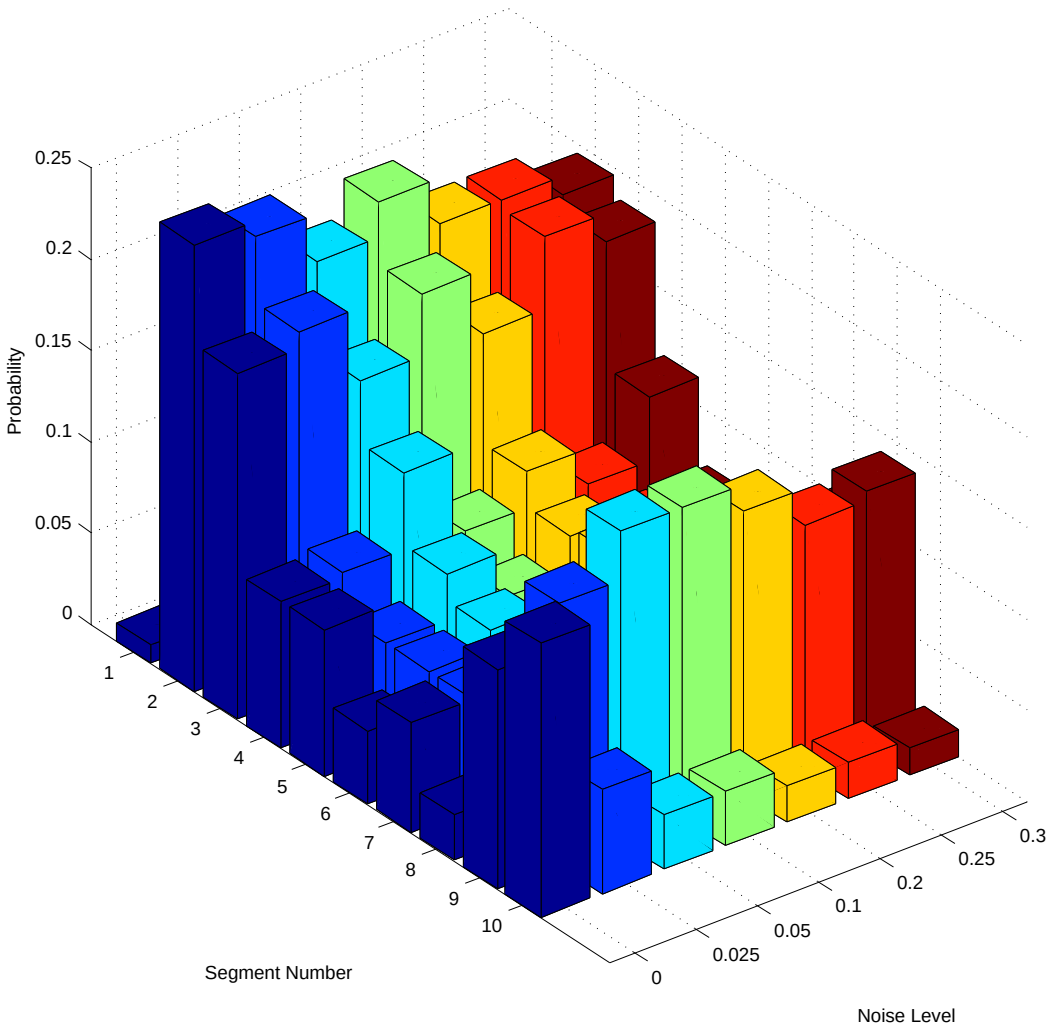

Figure 4.4. Resulting segment number histogram with noisy input.

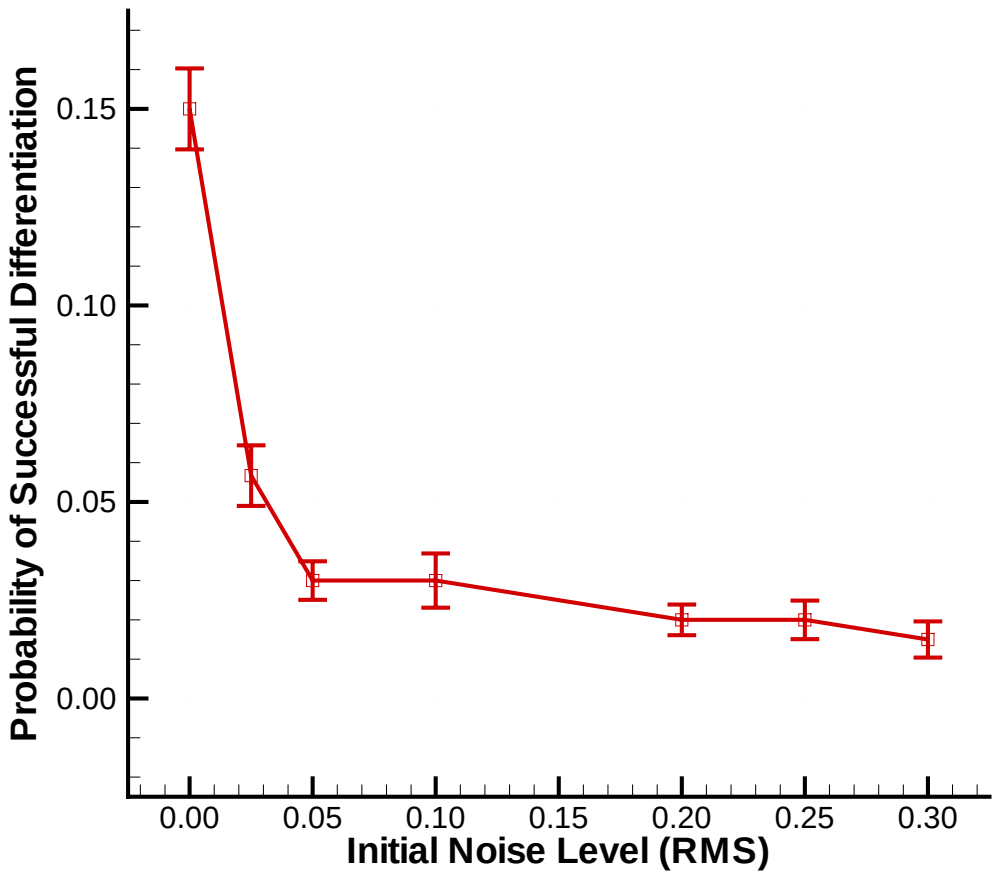

Figure 4.5. Differentiation response to input noise.

5. Simulating Tryptophan and Lactose Regulatory Networks

Biological networks are composed of a large number of interacting pathways. Developing a robust, efficient and accurate simulation of such large scale systems is a daunting and computationally intensive task. Similar to biological networks, electrical systems are composed of large numbers of subsystems and interacting subcircuits. Simulation of these systems must produce accurate results in an efficient manner. Parallel circuit simulation tools, such as *Xyce* (*http://www.cs.sandia.gov/xyce/*) allow engineers to model and test large scale systems. The theoretical framework used in developing *Xyce* also provides the necessary tools for constructing a biological circuit simulator which can model multivariate, multiscale, hybrid biological networks.

In order to take advantage of the *Xyce* simulation framework, we create circuit abstractions of biological elements and construct netlist files that are executed using *Xyce*. Similar to the abstractions used in flux balance analysis (FBA), the flow of metabolic and genetic substrates are synonymous to the flow of current through an electrical circuit [14, 15]. Interactions in genetic regulatory circuits were initially represented using CMOS-based logic sub-circuits. Boolean logic simulations of gene networks is consistent with the boolean network models for genetic regulation used by other researchers [16, 15]. Metabolic reactions are simulated using analog sub-circuits where metabolite accumulation and degradation are modeled using capacitors and resistors, respectively. A genetic clock (modeled as a discrete-value voltage source) is used to synchronize genetic events. The enzymatic products of genetic pathways function as “metabolic clocks” by controlling when metabolic reactions occur.

5.1 Application to *Escherichia coli* Regulatory Pathways

Using *Xyce*, we simulate the metabolic and genetic regulation of the tryptophan and lactose biosynthesis. These two pathways were selected as example systems for constructing and testing the circuit simulation framework because the genetic and metabolic networks directly interact. In both systems the final output of the metabolic circuit is an input into the gene regulatory network, which coordinates the expression of enzymes that catalyze the metabolic reactions.

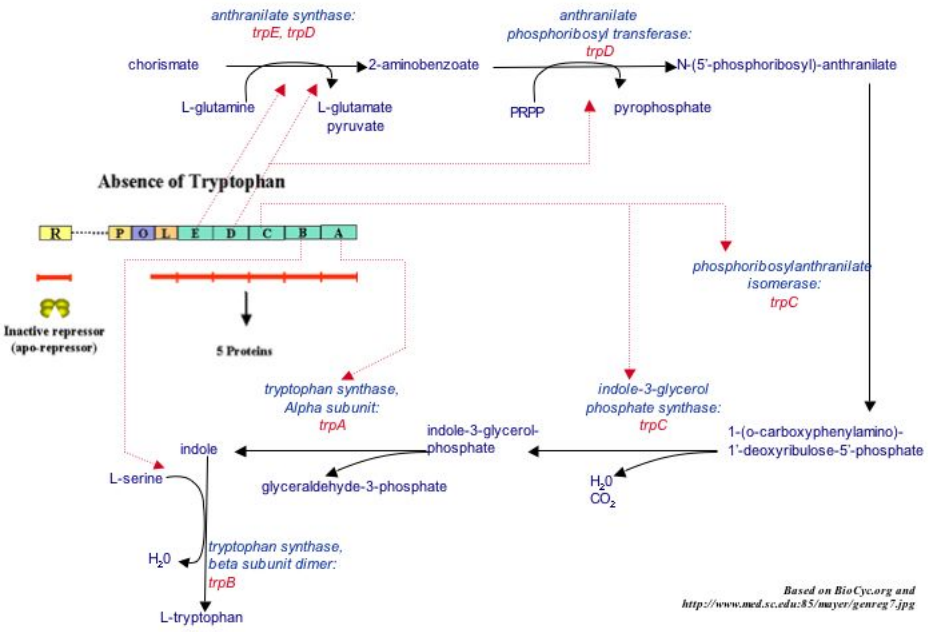

Figure 5.1. Genetic and metabolic control for the activation of tryptophan biosynthesis.

In bacteria, the synthesis of tryptophan is regulated primarily by feedback inhibition where the presence of tryptophan negates the production of four enzymes that catalyze various steps in tryptophan biosynthesis: anthranilate synthase, anthranilate phophoriosyl transferase, indole glycerol phosphate synthase, and tryptophan synthase [17]. The presence of tryptophan represses the transcription of the tryptophan operon. The operon contains five structural genes: trpE, trpD, trpC, trpB, and trpA. The repressor gene, trpR, the promoter region, the operator region and the trpL leader gene are also part of the the tryptophan operon (see Figure 5.1).

In the absence of tryptophan (or very low levels of tryptophan in the bacterial cell), the RNA polymerase molecule is able to transcribe the region of the DNA that codes for the trpEDCBA genes. The ribosome translates the resulting messenger RNA (mRNA) and produces the enzymes that catalyze the metabolic reactions for tryptophan biosynthesis (Figure 5.1). Once a significant level of tryptophan is present in the cell the tryptophan feeds back into the genetic regulation processes and repression occurs in two steps. Tryptophan, acting as a corepressor, binds to the inactive repressor protein (an aporepressor), which is the result of transcribing and translating gene trpR. The activated repressor/tryptophan complex, a holorepressor, binds to the tryptophan operon. This prevents the RNA polymerase from transcribing the trpEDCBA gene, hence the enzymes needed for catalysis of the metabolic

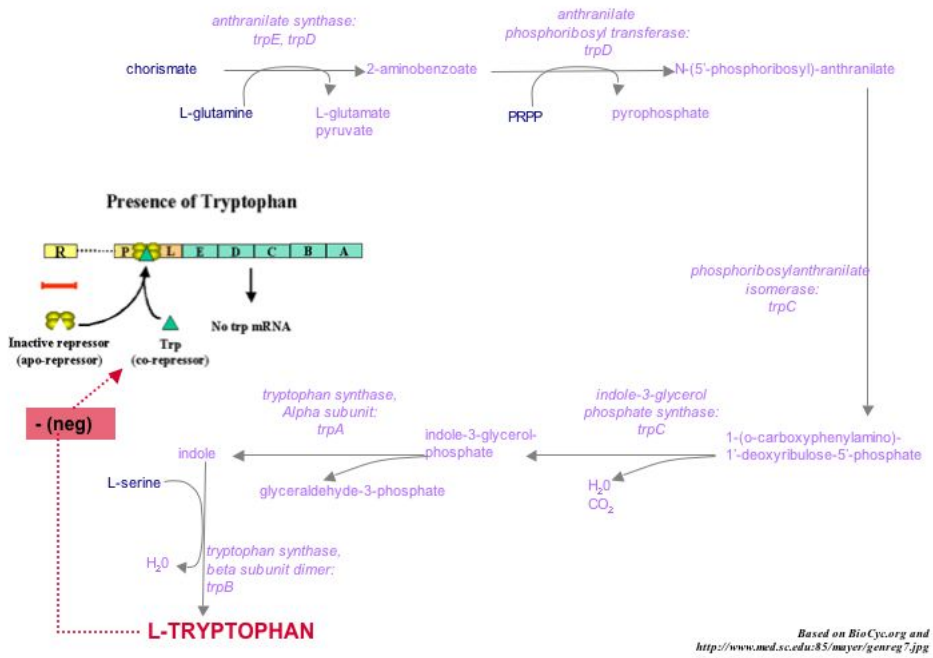

Figure 5.2. Genetic and metabolic control for the repression of tryptophan biosynthesis.

reactions in the biosynthesis pathway are not formed and tryptophan biosynthesis stops (Figure 5.2).

The genetic regulation, derived from Xiu et al.'s work [17], is modeled using Boolean circuits with tryptophan concentration as the only variable input and the presence (logic 1) or absence (logic 0) of trpEDCBA as the output. The current implementation leaves room for the incorporation of a more detailed genetic circuit model that takes into account factors such as mRNA concentration, genetic rate constants, and gene levels. The existing model makes the simplifying assumption that the presence of the trpEDCBA gene correlates to the successful production of the corresponding enzymes.

The stoichiometric reactions involved in tryptophan biosynthesis are (data from *http://www.ecocyc.org* and *http://gcrg.ucsd.edu/organisms/ecoli.html*):

$$\begin{gathered} chor + gln \rightarrow glu + pyr + an \\ an + prpp \rightarrow ppi + npran \\ npran \rightarrow cpad5p \\ cpad5p \rightarrow co2 + igp \\ igp + ser \rightarrow t3p1 + trp \end{gathered} \tag{5.1}$$

Table 5.1. Abbreviations for metabolites involved in tryptophan biosyntesis.

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

Table 5.1 lists the abbreviations for the metabolites involved and modeled in our Xyce simulation of tryptophan biosynthesis. To connect the gene network and the metabolic reaction network, we developed CMOS-based transmission gates and incorporated the gates into the stoichiometric reaction subcircuit. This permits the output of the gene network to regulate when and if a metabolic reaction occurs. Figure 5.3 depicts the tryptophan circuit. Stoichiometric data (from *http://gcrg.ucsd.edu/* and *http://biocyc.org/*) and qualitative descriptions available in literature [18, 17] are used to construct the coupled metabolic and genetic circuit netlist:

```
*****************************************************************************
Tryptophan Biosynthesis Circuit

**************** Genetic Circuit
* Initial conditions for logic circuit
** Set voltage and clock values
Vdddev nVdd 0 5V
VddNdev nVddN 0 -5V
Vset set 0 0V
Vreset reset 0 0V

Vclk genClk 0 PULSE(-5V 5V 0 0.1 0.1 .8 1.6)
xNeg_gclk genClk genClkNot neg

* Initial Conditions:  assume apoRep and mRNA available at constant concentration
```

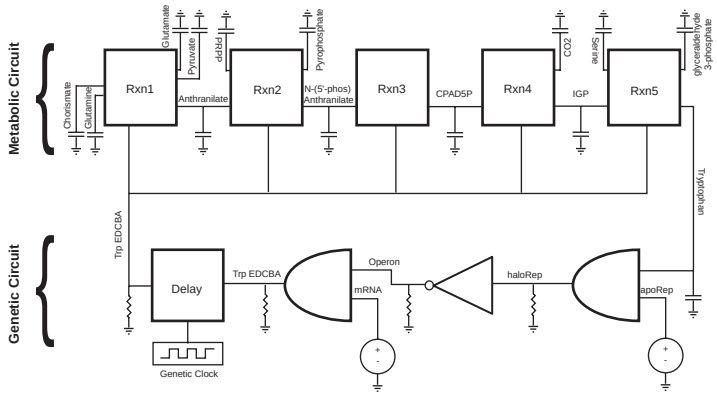

Figure 5.3. Circuit diagram of tryptophan genetic and metabolic control

```
*    [apoRep] = 10 M  10V=positive 1 in CMOS logic gates
*    [mRNA] = 10 M
Vin_apoRep apoRep 0 5V
Vin_mRNA mRNA 0 5V

xAnd_holoRep trp apoRep holoRep nVdd and2
xNot_opr holoRep opr nVdd not
xAnd_trpEDCBAprev opr mRNA trpEDCBAprev nVdd and2
xDff_trpEDCBA trpEDCBAprev enzymeIn enzymeInNot nVdd nVddN
  genClk genClkNot set reset dff
xEClk_trpEDCBA enzymeIn trpEDCBA trpEDCBANot enzymeClk
PARAMS: oldmaxVal=5 newmaxVal=5

RresHoloRep holoRep 0 100K
RresOpr opr 0 100K
RresTrp trp 0 100K
RresTrpEDCBAprev trpEDCBAprev 0 100K

********************** Metabolic Circuit
* Initial Conditions for metabolic network
*    [chor] = 0.01 M
*    [gln] = 0.01 M
*    [prpp] = 0.01 M
*    [ser] = 0.01 M
*
* Externally provided
CcapChor chor 0 1 IC=5
```

```
CcapGln gln 0 1 IC=5
CcapPrpp prpp 0 1 IC=5
CcapSer ser 0 1 IC=5

*   Internally produced but not consumed
CcapGlu glu 0 1 IC=.01
CcapPyr pyr 0 1 IC=.01
CcapPpi ppi 0 1 IC=.01
CcapCo2 co2 0 1 IC=.01
CcapT3p1 t3p1 0 1 IC=.01

* Internally produced and consumed in tryp operon genetic network
CcapTrp trp 0 1 IC=.01

* Internally produced and consumed in tryp metabolic network
CcapAn an 0 1 IC=.01
CcapNpran npran 0 1 IC=.01
CcapCpad5p cpad5p 0 1 IC=.01
CcapIgp igp 0 1 IC=.01

** chor + gln -> glu + pyr + an
xRxn_ChorGln_GluPyrAn chor gln glu pyr an trpEDCBANot trpEDCBA nVdd nVddN Rxn2To3
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 p3Stio=1 fRate=1
** an + prpp -> ppi + npran
xRxn_AnPrpp_PpiNpran an prpp ppi npran trpEDCBANot trpEDCBA nVdd nVddN Rxn2To2
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 fRate=1
** npran -> cpad5p
xRxn_Npran_Cpad5p npran cpad5p trpEDCBANot trpEDCBA nVdd nVddN Rxn1To1
PARAMS: r1Stio=1 p1Stio=1 fRate=1
** cpad5p -> co2 + igp
xRxn_Cpad5p_Co2Igp cpad5p co2 igp trpEDCBANot trpEDCBA nVdd nVddN Rxn1To2
PARAMS: r1Stio=1 p1Stio=1 p2Stio=1 fRate=1
** igp + ser -> t3p1 + trp
xRxn_IgpSer_T3p1Trp igp ser t3p1 trp trpEDCBANot trpEDCBA nVdd nVddN Rxn2To2
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 fRate=1

** Analysis
.TRAN 0.25s 20s 0
** Output
.PRINT TRAN V(genClk) V(holoRep) V(trpEDCBAprev) V(enzymeIn) V(trpEDCBA)
V(chor) V(an) V(glu) V(npran) V(cpad5p) V(igp) V(trp)
*****************************************************************************
```

Figure 5.4 shows the results of the tryptophan circuit simulation. In Figure 5.4 the horizontal axis is simulation time (tSim) and the vertical axis is concentration of the enzyme

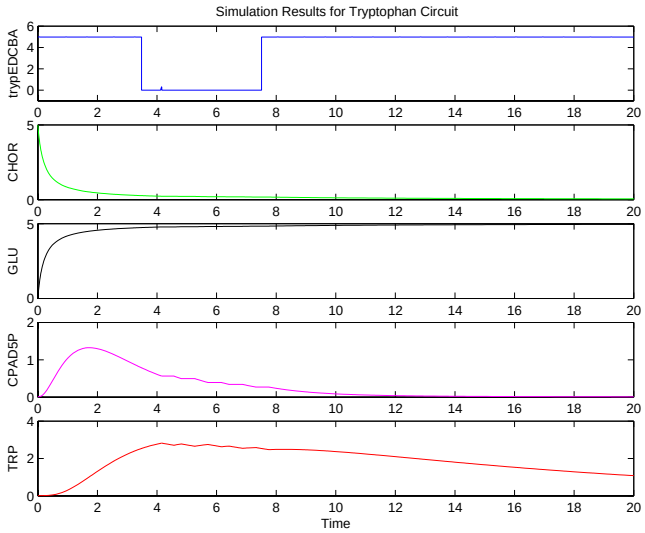

Figure 5.4. Circuit diagram of tryptophan genetic and metabolic control.

and metabolites involved. The simulation results illustrate features of the well known tryptophan regulation process. At simulation time tSim=0, tryptophan is not present in the system so the genetic circuit permits the transcription (and ultimately translation) of the trpEDCBA enzyme (first graph of Figure 5.4) which catalyzes reactions in the tryptophan biosynthesis pathway. As depicted in Figure 5.4, tryptophan (TRP; fifth graph) is produced as chorismate (CHOR; second graph) and 1-(o-carboxyphenylamino)-1'-deoxyribulose-5'-P (CPAD5P;
fourth graph) are consumed. While CPAD5P is produced and consumed during tryptophan biosynthesis, Glutamate (GLU; third graph) is produced but not consumed by any step in the biosynthesis pathway. Once a significant level of tryptophan is present in the system (e.g. simulation time tSim= 4) transcription of the trpEDCBA enzyme is inhibited. As tryptophan is degraded (tSim=4 to tSim= 7.5) transcription of the trpEDCBA gene resumes once tryptophan levels fall below threshold (currently the boolean logic sub-circuit is activated when levels fall below approximately 0.5*(Maximum Substrate Concentration)). After tSim=7.5, although tryptophan levels increase slightly, there are not enough reactants in the system to produce sufficient tryptophan levels in the presence of the degradation element.

Boolean Kinetics Model of Gene Regulation

Interactions in genetic regulatory circuits were initially represented using CMOS-based Boolean circuit. We found that as the network size increased (discovered for larger, eighty-one node gene network simulation), successful simulation became prohibitive. Therefore

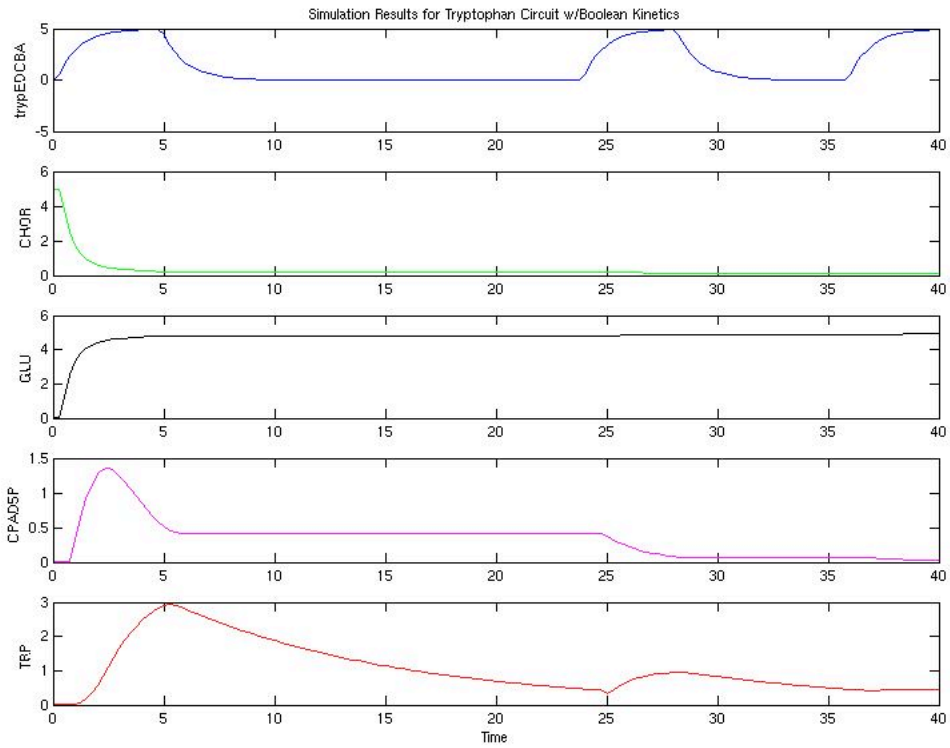

Figure 5.5. Simulation results of tryptophan genetic and metabolic circuit using a Boolean kinetics framework for gene network modeling.

we devised non-CMOS based logic gates using Boolean kinetics [19, 20]. In Boolean kinetics, gates are of the form:

$$\frac{dY}{dt} = F(X, T_Y) - aY$$

where a and T_Y are variable parameters and for our simulations $F(X)$ is a unit step function. Boolean kinetics allowed us to produce analog waveforms for the gene network and lends towards a more realistic depiction of genetic networks. The use of the Boolean kinetics framework for genetic circuits produced similar results as depicted in Figure 5.5.

5.2 Lactose Operon

The genes that code for the enzymes responsible for the metabolism of lactose, along with the operator and promoter regions (the operator and promoter are specific regions of the DNA that are next to the the protein coding genes) upstream of the genes form the lactose operon. The three structural genes, $lacZ$, $lacY$, and $lacA$ code for β-galactosidase, a permease, and galactose transacetylase, respectively. The current Xyce simulation of lactose metabolism only represents the enzymatic activity of the $lacZ$ gene, the β-galactosidase gene.

The stochiometric equation for lactose catabolization is

$$lac \rightarrow glac + glc \tag{5.2}$$

where lac is Lactose, $glac$ is Galactose, and glc is a-D-Glucose. The production of the $lacZYA$ gene is regulated by the elements on both sides of the stochiometric relation. A repressor gene (designated as $LacRep$ in the lactose circuit diagram, Figure 5.6) codes for a repressor protein that binds to the lactose operator region. The binding of the repressor protein interferes with and inhibits the transcription of $lacZYA$ and the production of the enzymes needed for lactose metabolism. When an inducer protein such as lactose is present in the cell, the repressor protein is unable to bind to the operator and transcription proceeds successfully. The amount of glucose in the cell also affects the expression of the lactose genes, this effect is called catabolite repression. If glucose and an inducer are both present, the operon will not be fully expressed. When the levels of glucose are reduced, the levels of cAMP increase. cAMP binds to a protein called CAP (also called CRP) forming a complex that binds to the promoter region, increasing the efficiency of transcription initiation. In the Boolean model of the genetic regulatory circuit, the effect of the inducer plus the effect of glucose is modeled as an AND event. Figure 5.6 shows a schematic of the simulated lactose metabolic and gene regulatory network. As in the tryptophan circuit, we use stoichiometric data (http://gcrg.ucsd.edu/ and http://biocyc.org/) and qualitative descriptions [18] to construct the coupled metabolic and genetic circuit netlist for the lactose operon:

```
*************************************************************************
Lactose Degradation Circuit
******************************** Genetic Circuit
* Initial conditions for logic circuit
** Set voltage and clock values
Vdddev nVdd 0 5V
VddNdev nVddN 0 -5V

Vset set 0 0V
Vreset reset 0 0V

Vclk genClk 0 PULSE(0V 5V 0 0.1 0.1 .8 1.6)
```

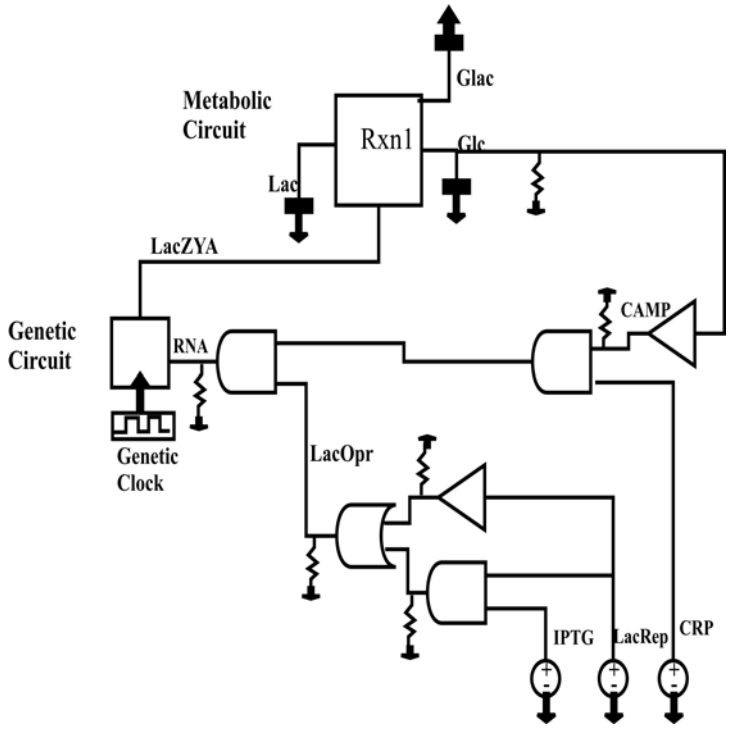

Figure 5.6. Circuit diagram of lactose genetic and metabolic control

```
*xNeg_gclk genClk genClkNot neg
xNot_clk genClk genClkNot nVdd not


* Initial Conditions:  assume lacRep, CRP  available at constant concentration
*    [lacRep] = 5 M  5V=positive 1 in CMOS logic gates
*    [CRP] = 5 M  5V=positive 1 in CMOS logic gates
* glc=glucose   glac=galactose  lac=lactose iptg=isopropylthiogalactosidase
Vin_lacRep lacRep 0 5V
Vin_crp crp 0 5V
Vin_iptg iptg 0 5V

*Rep-Inducer subcirc - Operator circ
xNot_lacRepNot lacRep lacRepNot nVdd not
xAND_lacRepIptg lacRep iptg lacRepIptg nVdd and2
xOr_lacOpr lacRepNot lacRepIptg lacOpr nVdd or2

* Transcrip/Translat Circ
xNot_cAMP glc cAMP nVdd not
```

```
xAND_cAMPcrp cAMP crp cAMPcrp nVdd and2
xAnd_mRNA cAMPcrp lacOpr mRNA nVdd and2
xEClk_lacZYA mRNA lacZYA lacZYANot enzymeClk PARAMS: oldmaxVal=5 newmaxVal=5

RresLacRepNot lacRepNot 0 100K
RresLacRepIptg lacRepIptg 0 100K
RresLacOpr lacOpr 0 100K
Rres_cAMP cAMP 0 100K

*RresGlc glc 0 100K
** Lower resistance to simulate degradation
RresGlc glc 0 10
Rres_mRNA mRNA 0 100K

********************************** Metabolic Circuit
* Initial Conditions for metabolic network
*    [lac] = 0.01 M
* Externally provided
CcapLac lac 0 1 IC=5

*   Internally produced but not consumed
CcapGlac glac 0 1 IC=.01

* Internally produced and consumed in tryp operon genetic network
CcapGlc glc 0 1 IC=.01

** lac -> glc + glac
xRxn_lac_GlcGlac lac glc glac lacZYANot lacZYA nVdd nVddN Rxn1To2
  PARAMS: r1Stio=1 p1Stio=1 p2Stio=1 fRate=1


** Analysis
.TRAN 1s 20s 0
** Output
.PRINT TRAN V(genClk) V(lac) V(glc) V(glac) V(mRNA) V(lacZYA) V(cAMPcrp)


*****************************************************************************
```

Lactose simulation results, Figure 5.7 correspond to qualitative descriptions of the operon model. The results represent a circuit without the delay element or genetic clock. At simulation time tSim=0, lactose (LAC) is transported into the system. The absence of glucose (GLC) in the system causes the genetic circuit to permit the production of the β-galactosidase (lacZYA) enzyme which breaks down lactose into glucose and galactose (GLAC). Glucose is produced and degraded in this circuit. Once glucose levels reach a

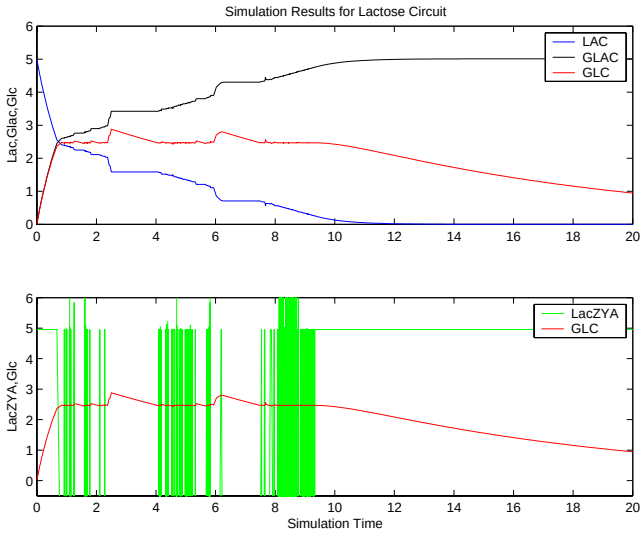

Figure 5.7. Circuit diagram of lactose genetic and metabolic control

threshold point, the genetic circuit turns off the production of lacZYA (graph 2 in Figure 4). Note, galactose is produced but not degraded. Between tSim=1 and tSim=2 the level of glucose oscillates due to the switching on and off of the lacZYA enzyme. This behavior can also be observed for tSim=4 to 6 and tSim= 7.7 to 9.4. After tSim= 9.5, although glucose levels are below threshold and lacZYA is present, lactose levels are too low to produce sufficient amounts of glucose to overcome the degradation element (graph 1 of Figure 4).

The model elements and framework developed and refined for the tryptophan and lactose systems are used to implement large-scale Xyce simulations of whole-organisms.

6. Simulation of an Eighty-One Node Inferred Gene Network

6.1 Boolean Logic Framework for Simulation

Quantitative descriptions of biochemical pathways that govern the functional behavior of cells are constructed using computational frameworks such as mathematical logic and Boolean algebra, differential equations, and graph theory [Liang et al. 1998; Smolen et al., 2000; Dutilh 1999; Thiefry and Thomas, 1998; McAdams and Shapiro, 1995; McAdams and Arkin, 1998; Covert, et al. 2001, Schilling, et al. 2000, Wiechert 2002; Wagner and Fell, 2001]. The question the model attempts to answer determines the mathematical framework used. For instance, models used in the analysis of metabolic pathways typically employ differential equations in order to recover parameters useful in predicting growth and by-product secretion rates [Covert, et al. 2001, Schilling, et al. 2000]. But when the model is used to determine the qualitative behavior of a pathway, such as gene regulatory pathways [Thieffry and Thomas, 1998], a logical representation of the system may be a necessary and sufficient simplification. To construct a whole-cell gene network simulation we use mathematical logic and Boolean algebra as the framework for constructing our model.

Mathematical Logic and Boolean Algebra

The field of Computer Engineering uses mathematical logic in the design of digital systems for computer hardware. A digital system is a system with (1) Discrete (usually 0 and 1 in a binary digital system) inputs and outputs; (2) Explicit functions that translate the input stream into new output streams [Katz 1994]. Digital system design is the second level in the design of complex hardware systems. Complex hardware systems can be described using three levels of abstraction (each level representing increased details): system, logic, and circuit levels. System level describes the input and output behavior in an abstract manner using flowcharts or computer programs; logic level uses logic gates as building blocks to implement the behavior of underlying components of the system; circuit level uses electrical components, such as resistors and capacitors, to implement the building blocks used in the logic level [Katz 1994]. Figure 6.1 parallels these hardware system abstraction levels to biological system description levels.

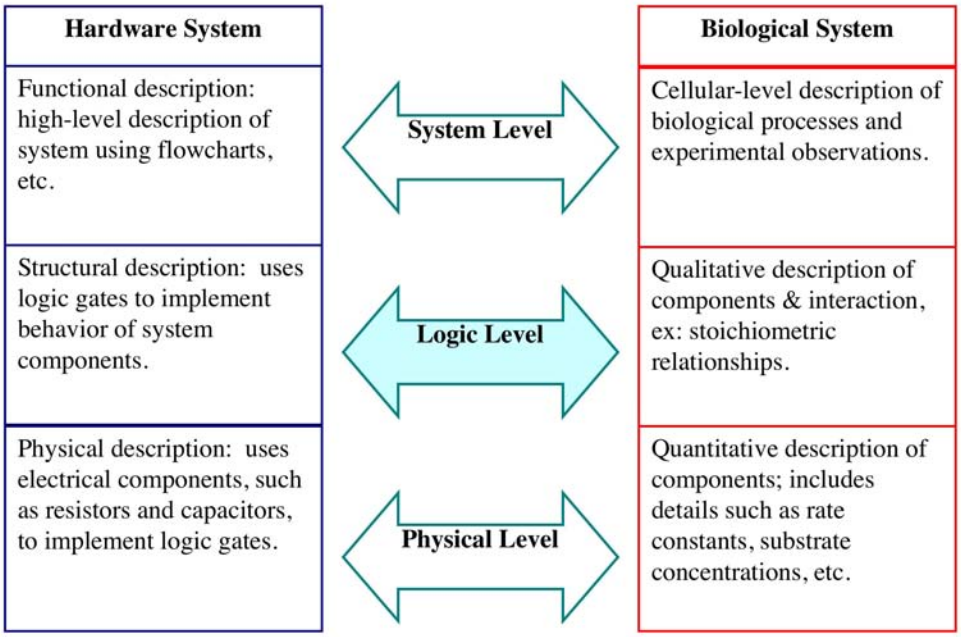

Figure 6.1. Biological parallels to hardware system abstraction layers.

There are several ways to represent a system using mathematical logic (permits us to categorically evaluate the truth of a set of statements) and Boolean algebra (algebraic method for combining logic values). Possible system representation includes: [Katz 1994].

- Truth table: a list of all possible input combinations and the resulting output values for the given system.
- Logic gates (schematic representation): implementation of the system using interconnected primitive components (logic gates, ex: OR, AND, NOT)
- Boolean equations: an algebraic short-hand for representing the truth table of a function or system.

Combinational logic units, are digital systems that can be defined as a network of interconnected gates and switches without feedback, Figure 6.2(B). Their output depends only on the current inputs. Sequential logic circuits incorporate feedback; their output depends on the current input and the history of previous inputs or the previous state of the system [Katz 1994]. Returning again to Kaufman, et al.s 1985 B-cell/T-cell model, the updating function for the helper T-cell, H, is defined as: H = e * s + h, (where s indicates negation of s). Figure 6.3 shows the logic gate schematic.

6.2 Modeling Gene Networks with Boolean Logic

The use of mathematical logic to quantify biological processes began in the late nineteen sixties with Kaufmans work which uses Boolean networks to analyze biochemical pathways and model interactions of immune molecules [Dhaeseleer, 2000; Pereleson and Weisbuch, 1997]. Boolean networks model each node or gene in the case of gene networks as on (a value of 1) or off (a value of 0). The updating function for each node is dependent on the current state of all its k input nodes [Dutilh, 1999; Dhaseleer, 2000]. Although Boolean networks are a gross approximation of the intricate interactions present in biochemical networks or immune response networks, they have yielded useful results in the analysis or experimental gene expression data and for reverse engineering of biological networks [Dhaeseleer, 2000].

Circuit Constructs for Large-Scale Gene Network Modeling

In order to accomplish our goal of hybrid modeling by coupling genetic and metabolic network simulations, a focus of this project was to develop primitives for modeling gene networks. Gene networks are modeled as Boolean networks using logic gates (AND, OR, NOT). We initially developed CMOS based logic gates and delay elements for gene level simulation of the tryptophan and lactose operon networks. The CMOS based models were

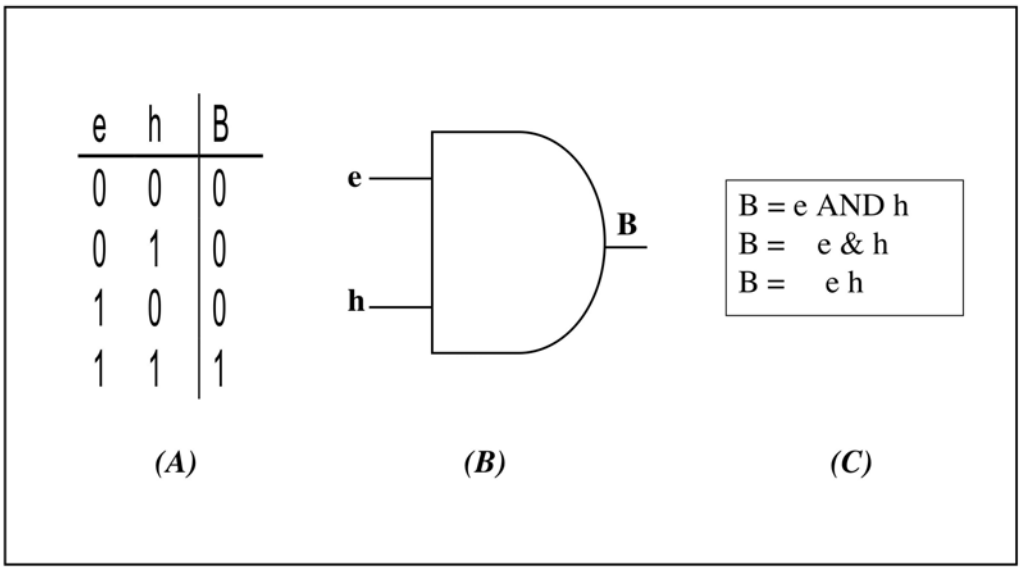

Figure 6.2. Truth table (A), logic gate schematic (B), and Boolean equation (C) representation of B-cell population (from Kaufman, Urbain, and Thomas (1985) B-cell/T-cell discrete interaction model [Perlesnon and Weisbuch, 1997]).

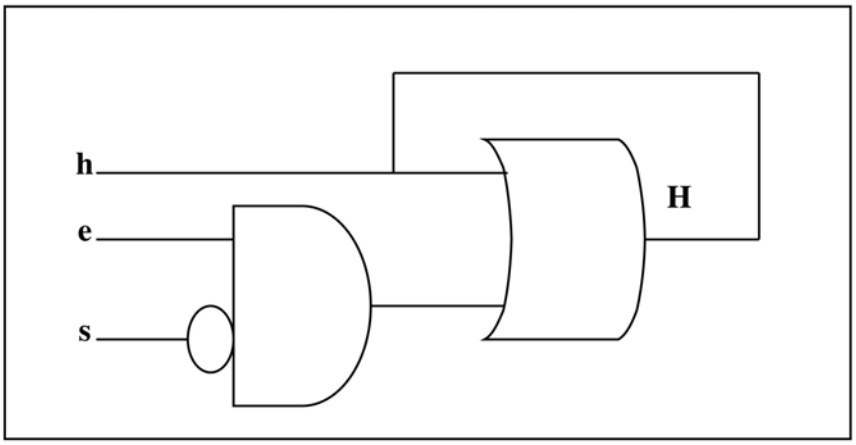

Figure 6.3. Logic gate schematic representation of helper T-cell population (from Kaufman, Urbain, and Thomas (1985) B-cell/T-cell discrete interaction model [Perleson and Weisbuch 1997].

Table 6.1. Boolean kinetics equations for implementation of genetic logic gates.

Logic Relation	Boolean Kinetics Equation
NOT (X)	$\frac{dY}{dt} = F(V_{Max} - X, T_Y) - aY$
X_1 AND X_2	$\frac{dY}{dt} = F(X_1, T_{Y,1}) * F(X_2, T_{Y,2}) - aY$
X_1 OR X_2	$\frac{dY}{dt} = F((F(X_1, T_{Y,1}) + F(X_2, T_{Y,2})), T_{Y,OR}) - aY$

used to successfully simulate small hybrid systems in *E. coli*. As the gene network size increased, causing the number of CMOS gates to also increase, the Xyce simulation was not feasible.

To accommodate whole organism gene networks, we developed logic gates using Boolean kinetics. AND, OR, and NOT gates are simulated using a variation of the dynamic boolean equations used in Shen-Orr, et al. 2002 paper which in turn is based on the Boolean kinetics equation presented in McAdams and Arkin, 1998. The general equation for a gene Y whose output concentration is dependent on the concentration of an input gene X:

$$\frac{dY}{dt} = F(X, T_Y) - aY \tag{6.1}$$

where in our implementation $F(X, T_Y)$ is a step function dependent on the activation threshold value, T_Y. The constatnt a is related to the degradation rate of Y. Variations of the general form, Equation 6.1, are used to represent the logic gates as depicted in Table 6.1: Below is the netlist implementation of the basic single input and double input Boolean kinetics logic gates represented in Table 6.1:

```
************** Boolean Kinetics Logic Gates********************

*******************************************************************

***** NOT Subcircuit
.SUBCKT NOT in1 outF PARAMS: koutF=1
***SUBCIRCUIT OR - Dynamic Boolean OR gate
*** Equation:  outF = NOT in1=> d outF/dt = F(Vmax - in1,TNot)- kout*outF
*** Note:  Vmax=Max voltage, TNot=0.5
.PARAM vMax=1
.PARAM TNot=0.5
BoutF outF 0 V={SDT(U(vMax-V(in1) - TNot) - V(outF)*koutF)}
.ENDS

*******************************************************************
***** AND Subcircuit
.SUBCKT AND in1 in2 outF PARAMS: T1=0.5 T2=0.5 koutF=1
```

```
*** SUBCIRCUIT AND - Dynamic Boolean AND gate
*** Equation:  outF = in1 AND in2 => d outF/dt = F(in1,T1)*F(in2,T2)
- kout*outF
BoutF outF 0 V={SDT(U(V(in1) - T1)*U(V(in2) - T2)-V(outF)*koutF)}
.ENDS
*******************************************************************

***** OR Subcircuit
.SUBCKT OR in1 in2 outF PARAMS: T1=0.5 T2=0.5 koutF=1
*** SUBCIRCUIT OR - Dynamic Boolean OR gate
*** Equation:  outF = in1 AND in2 => d outF/dt = F[(F(in1,T1) +
F(in2,T2)),TOR] - kout*outF
*** Note:  TOR=0.5
.PARAM TOR=0.5
BoutF outF 0 V={SDT(U((U(V(in1) - T1) + U(V(in2) - T2)) - TOR)
- V(outF)*koutF)}
.ENDS
*******************************************************************
```

6.3 Modeling and Simulation of Inferred Genetic Networks

We produced a logic circuit schematic for the inferred Boolean Networks. The network topology specified by the truth tables was used to produce the circuit netlist. Boolean kinetics-based logic components (AND-, OR-, NOT-Gates) are dynamically implemented to accommodate variable numbers of gene inputs (A. S. Sedra and K. C. Smith, 1991). The initial genetic logic circuit was visualized using the ChileCAD Schematic Capture tool. We simulate the Boolean circuit using the Xyce^{TM} Parallel Electronic Simulator. Simulation allowed us to capture the time-varying response of the network and compare the response of the inferred network to the microarray data. The goal is to develop a circuit schematic that can reproduce the experimental data.

Simulation of Whole-Cell Gene Network

We used the Boolean kinetics framework to simulate the complete inferred gene network of yeast. Time series microarray data for the yeast system was clustered into gene groups, which are referred to a meta-genes. The time-dependent expression profile of each meta-gene was discretized and served as the input to the genetic network inference algorithm [21]. The algorithm produces a truth table that includes every possible input/output relationship for the node. Each meta-gene node, Y, will have 2^k entries, where k is the number of meta-gene nodes that influence the output of Y. As the k increases the size of the truth table grows exponentially and the number of elements in the circuit grows substantially.

Table 6.2. Example reduction of meta-gene three using Espresso.

Espresso Input	Espresso Output
.i 4	.ilb n66 n44 n1 n0
.o 1	.ob n3
.ilb n66 n44 n1 n0	.i 4
.ob n3	.o 1
.p 6	.p 3
1000 1	0110 1
1010 1	10– 1
0110 1	-011 1
1001 1	.e
0011 1	
1011 1	
.e	

$$n3 = \overline{n66} * n44 * n1 * \overline{n0} + n66 * \overline{n44} + \overline{n44} * n1 * n0$$

Since the truth table representation is exhaustive and most likely redundant, simulation of the complete truth table would be wasteful. We can apply rules from Boolean algebra to produced a minimized representation of the inferred gene network. The minimized form contains all of the behavioral relationships and leads to a simpler circuit realization.

We perform a two-level Boolean minimization on the truth table representation of the inferred gene network using Espresso, a well-known logic simplification tool available from the University of California, Berkeley (www-cad.eecs.Berkeley.edu/software/software.html) [R. Katz, 1994]. We developed a software tool that receives a text file containing the inferred network. The truth table specification of each node is reformatted into a separate input file readable by Espresso. Espresso receives the encoded truth table representation of each node and produces a minimized truth table file. The number of product terms used to specify each node is minimized (i.e. the number of AND gates used to realize the Boolean function is reduced). We extract the resulting Boolean equation for each node from the Espresso output file. Each of the nodes in the gene network is processed in this manner, yielding a minimized representation of the gene network. Table 6.2 demonstrates the minimization process for meta-gene three (reduced from six to four product terms). It lists the Espresso input for the logic one elements of the inferred truth table, the minimized Espresso output file, and the resulting Boolean equation, which is simulated using Xyce. Larger reductions occur for several meta-genes (e.g., meta-gene forty-six is reduced from thirty-one to fourteen product terms and meta-gene seventy-one is reduced from thirty-six to sixteen product terms).

We developed a Perl script, $gene2cir.pl$, for automatically generating Xyce equivalent cir-

cuits from the truth table representation of the inferred gene network. *Gene2cir.pl* takes a formatted text file that represents a genetic network, reduces each meta-gene node using Espresso, and produces the corresponding netlist for the reduced, inferred gene network. The below excerpt from the whole-cell yeast gene network simulation shows the netlist entries for meta-gene forty-four (Figure 6.5 includes a schematic of meta-gene forty-four).

```
*****************************************************************************
*Meta-gene 44 relevant netlist entries
********** NODE n0 *****
***** AND GATES FOR PRODUCT TERMS *****
xAnd_n0_0 n12in and_n0_0 and1
Rres_and_n0_0 and_n0_0 0 100K

***** OR GATE FOR SUM OF PRODUCT TERMS *****
xOr_n0 and_n0_0 n0 or1
Rres_n0 n0 0 100K

********** NODE n15 *****
***** AND GATES FOR PRODUCT TERMS *****
xAnd_n15_0 n47in_not n13in n0in_not and_n15_0 and3
Rres_and_n15_0 and_n15_0 0 100K
xAnd_n15_1 n13in_not n0in and_n15_1 and2
Rres_and_n15_1 and_n15_1 0 100K

***** OR GATE FOR SUM OF PRODUCT TERMS *****
xOr_n15 and_n15_0 and_n15_1 n15 or2
Rres_n15 n15 0 100K

********** NODE n44 *****
***** AND GATES FOR PRODUCT TERMS *****
xAnd_n44_0 n70in_not n0in_not and_n44_0 and2
Rres_and_n44_0 and_n44_0 0 100K
xAnd_n44_1 n15in n0in and_n44_1 and2
Rres_and_n44_1 and_n44_1 0 100K

***** OR GATE FOR SUM OF PRODUCT TERMS *****
xOr_n44 and_n44_0 and_n44_1 n44 or2
Rres_n44 n44 0 100K

********** NODE n70 *****
***** AND GATES FOR PRODUCT TERMS *****
xAnd_n70_0 n36in n16in_not n2in_not n1in_not and_n70_0 and4
Rres_and_n70_0 and_n70_0 0 100K
xAnd_n70_1 n36in n16in n2in n0in_not and_n70_1 and4
Rres_and_n70_1 and_n70_1 0 100K
```

```
xAnd_n70_2 n36in_not n2in n1in_not n0in and_n70_2 and4
Rres_and_n70_2 and_n70_2 0 100K
xAnd_n70_3 n36in_not n16in_not n1in and_n70_3 and3
Rres_and_n70_3 and_n70_3 0 100K
xAnd_n70_4 n36in n16in n1in and_n70_4 and3
Rres_and_n70_4 and_n70_4 0 100K

***** OR GATE FOR SUM OF PRODUCT TERMS *****
xOr_n70 and_n70_0 and_n70_1 and_n70_2 and_n70_3 and_n70_4 n70 or5
Rres_n70 n70 0 100K

***** Voltage Input Values *****
Vn0init n0init 0 PULSE(0V 1V 0 0.05 0.05 1.8 1799)
xOr_n0init n0init n0 n0in or
Rres_n0init n0init 0 100
Rres_n0in n0in 0 100K
xNOT_n0in n0in n0in_not not
Rres_n0in_not n0in_not 0 100K

Vn15init n15init 0 PULSE(0V 0V 0 0.05 0.05 1.8 1799)
xOr_n15init n15init n15 n15in or
Rres_n15init n15init 0 100
Rres_n15in n15in 0 100K

Vn44init n44init 0 PULSE(0V 0V 0 0.05 0.05 1.8 1799)
xOr_n44init n44init n44 n44in or
Rres_n44init n44init 0 100
Rres_n44in n44in 0 100K
xNOT_n44in n44in n44in_not not
Rres_n44in_not n44in_not 0 100K

Vn70init n70init 0 PULSE(0V 1V 0 0.05 0.05 1.8 1799)
xOr_n70init n70init n70 n70in or
Rres_n70init n70init 0 100
Rres_n70in n70in 0 100K
xNOT_n70in n70in n70in_not not
Rres_n70in_not n70in_not 0 100K

*************************************************************************
```

Initial node values are inputs into $gene2cir.pl$. Nodes are asynchronously updated.

We use Xyce to simulate the eighty-one node (each node is a meta-gene) yeast gene network. Results were compared to the original discretized signal. Figure 6.4 shows the simulation output for meta-gene zero compared to the discretized microarray time series

expression data. The two graphs are similar, which is not surprising given that meta-

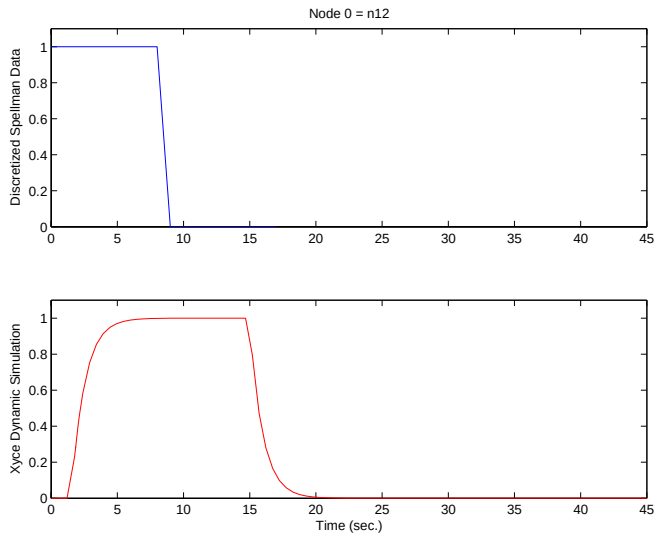

Figure 6.4. Comparison of the output of Xyce simulation (blue, top curve) to discretized time series microarray data (red, bottom curve) for meta-gene node zero.

gene zero has a simple gene network. A noticeable difference is the width of the two curves, which can be addressed by adjusting the threshold point and parameters used in the Boolean kinetics model of the logic gates. Figure 6.5 shows the simulation output of meta-gene node forty-four and its corresponding input nodes and a comparison of the Xyce simulation to the discretized microarray data. For this more complex example, it is difficult to find similarities between the nodes (except for node zero, previously discussed). We suspect that some of the observed variations between the simulation and discretized data are due in part to the fact that the discretized data represents eighteen time points whereas the simulation data represents multiple time steps. Application of a parameter estimation tool, like $Dakota^{TM}$, will help tune simulation parameters to produce results consistent with the discretized microarray data.

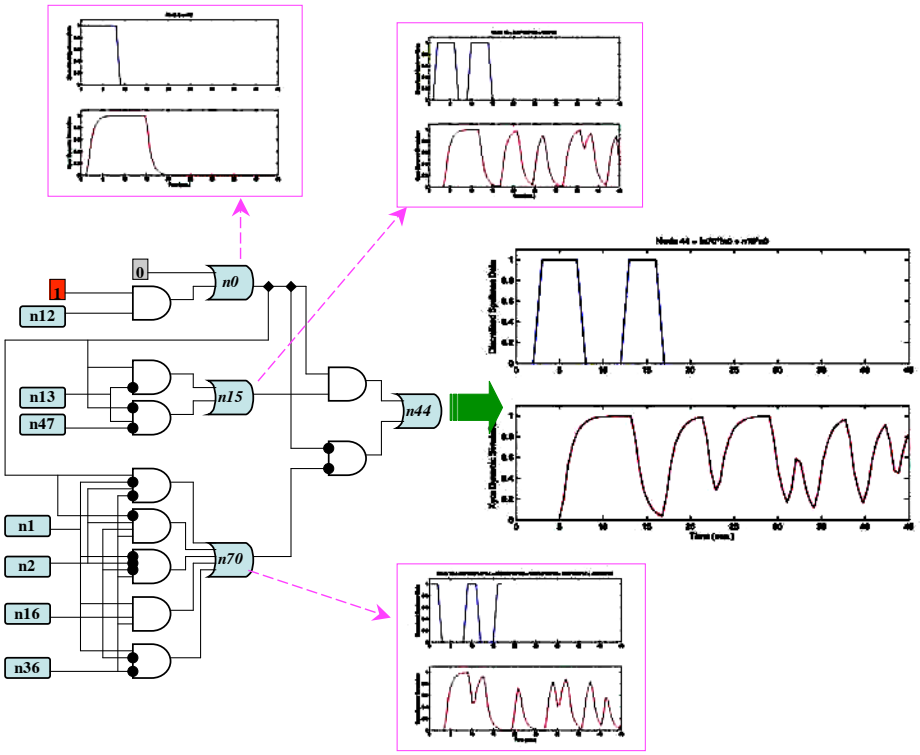

Figure 6.5. Comparison of Xyce simulation results (blue, top graphs) for meta-gene Node 44 and Nodes 0, 15, and 70 (inputs to Node 44) to discretized microarray data (red, bottom graphs).

7. Simulating *Escherichia coli* K-12

A major goal of this work is to demonstrate the feasibility of implementing a Xyce simulation of a whole cell. Bacteriophage-λ was our original target organism. We modified our target system to the *Escherichia coli* K-12 due to the availability of metabolic and gene regulatory data. The whole cell model was constructed using the building blocks tested in the smaller tryptophan and lactose systems.

7.1 The Whole-Cell Metabolic Model

Circuit Components for Whole-Cell Model

As in the tryptophan circuit models, we use capacitors coupled to resistors to store produced metabolites and dissipate consumed metabolites. Using a generalized, Michales-Menten type equation, we use Xyce's analog behavioral modeling element, the $BSOURCE$, to model each reaction as:

$$Rate = C_M \times fRate \times (S_1^{C_1}) \times (S_2^{C_2})$$

Reactions can be unidirectional or bidirectional. Unidirectional reactions are implemented with a single $BSOURCE$ element while bidirectional, or reversible reactions, are implemented with two $BSOURCE$ elements. For example, the following reaction is bidirectional:

$$G6P + NADP \leftrightarrow D6PGL + NADPH \tag{7.1}$$

Table 7.1 lists the abbreviations for the metabolites involved. To implement we divide the above into two reactions:

$$\begin{aligned} G6P + NADP &\rightarrow D6PGL + NADPH \\ D6PGL + NADPH &\rightarrow G6P + NADP \end{aligned} \tag{7.2}$$

Figure 7.1 shows the circuit schematic for the reversible reaction. The corresponding netlist is replicated below.

```
xRxn_G6PNADP_D6PGLNADPH G6P NADP D6PGL NADPH
   enzyme enzymeNot nVdd nVddN Rxn2To2
```

Table 7.1. Abbreviations for metabolites involved in example reversible reaction.

Abbreviation	Metabolite
G6P	Glucose 6-phosphate
NADP	Nicotinamide adenine dinucleotide phosphate
D6PGL	D-6-Phosphate-glucono-delta-lactone
NADPH	Dihydronicotinamide adenine dinucleotide phosphate reduced

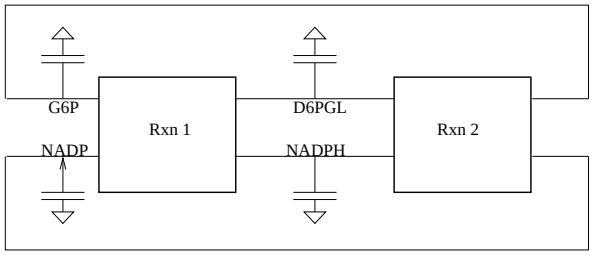

Figure 7.1. Circuit schematic for a bidirectional metabolic reaction.

```
   PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 fRate=0.001
** REVERSIBLE
xRxn_D6PGLNADPH_G6PNADP D6PGL NADPH G6P NADP
   enzyme enzymeNot nVdd nVddN Rxn2To2
   PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 fRate=0.001
```

We implement the two-substrate/two-product metabolic subcircuit as follows:

```
**** Rxn2To2 ****
.SUBCKT Rxn2To2 rnt1 rnt2 prd1 prd2 enzyme enzymeNot nVdd nVddN
   PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 fRate=1.67e-5

.PARAM CapReact1=1
.PARAM CapReact2=1
*** Enzyme activity as a Tx switch
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
xtx_rnt2 rnt2 rnt2in enzymeNot enzyme nVdd nVddN tx
Rres_rnt2in rnt2in 0 100K

*** Reaction - consume the reactants
```

```
BReact1 rnt1 0 I={r1Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) )  *
   ( (CapReact2 * V(rnt2in))**(r2Stio) ) }
BReact2 rnt2 0 I={r2Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) )  *
   ( (CapReact2 * V(rnt2in))**(r2Stio) ) }

*** Reaction - produce the products
BProd1 0 prd1 I={p1Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) )  *
   ( (CapReact2 * V(rnt2in))**(r2Stio) ) }
BProd2 0 prd2 I={p2Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) )  *
   ( (CapReact2 * V(rnt2in))**(r2Stio) ) }
.ENDS
```

As in the tryptophan and lactose circuit models, we use a transmission gate to simulate the enzymatic control of metabolic processes. The enzymatic transmission gate is implemented using a PMOS and NMOS with model parameters as shown in the netlist below.

```
***** Tx Gate Subcircuit
.SUBCKT TX inA outF clkL clkH nVdd nVddN
*** SUBCIRCUIT TRANSMISSION GATE
*** Equation:  outF = inA (clkH=clk, clkL=NOT(clk))
Mp1 outF clkL inA nVdd cd4012_pmos l=5u w=270u
Mn1 inA clkH outF nVddN cd4012_nmos l=5u w=175u
.ENDS

*******************************************************************
.MODEL cd4012_pmos PMOS (
+ LEVEL = 3  UO = 310  VTO = -1.6  NFS = 5.794E+10
   TOX = 6E-08  NSUB = 5.701E+15
+ NSS = 0    VMAX = 5.374E+04   RS = 5.359   RD = 93.66
   RSH = 0   IS = 1E-14
+ XJ = 7.9E-06   LD = 3E-08   DELTA = 0   THETA = 0.0278
   ETA = 0.535   KAPPA = 0.643
+ KP = 1.711E-05 L=5u W=270u GAMMA=0.37 PHI=0.65
   NFS=1E10
+ CBD=0.1P CBS=0.1P PB=0.81 CGSO=2P CGBO=4P CGDO=2P
   CJ=2E-4 MJ=0.5 CJSW=1E-9
+ MJSW=0.5 JS=1E-8 TPG=0 KF=1E-25 AF=1 FC=0.5 TNOM=27)
*************************
.MODEL cd4012_nmos NMOS (
+ LEVEL = 3 UO = 190   VTO = 1.679   NFS = 2.368E+11
   TOX = 6E-08   NSUB = 8.601E+15
+ NSS = 0   VMAX = 4.206E+04   RS = 13.21   RD = 11.59
   RSH = 0   IS = 1E-14
+ VMAX = 4.206E+04 NFS=1E10  GAMMA=0.37 PHI=0.65
+ XJ = 7.1E-06   LD = 8.6E-07   DELTA = 0   THETA = 0.0021
```

```
   ETA = 0.057   KAPPA = 0.15
+ KP = 2.161E-05  L=5u W=175u
+ CBD=0.1P CBS=0.1P PB=0.81 CGSO=2P CGBO=4P CGDO=2P
   CJ=2E-4 MJ=0.5 CJSW=1E-9
+ MJSW=0.5 JS=1E-8 TPG=0 KF=1E-25 AF=1 FC=0.5 TNOM=27)
```

Constructing the $E.coli$ Metabolic Netlist

Our first model was a metabolism only model. We use stoichiometric data to generate the wiring schematic for the whole-cell circuit. The stoichiometric data for *E. coli* metabolism is from the website maintained by Bernhard Palsson's lab at the University of California San Diego(*http://gcrg.ucsd.edu/*). The original Excel spreadsheet was formatted and converted to a text, comma-delimited file. This served as the input to $path2cir.pl$, a Perl script we developed for automatically generating the Xyce equivalent metabolic circuit from stoichiometric data. Due to the number of reactions involved, a method for automatically generating the netlist is necessary and will help reduce input error. In addition to constructing an automated netlist generator for delimitted text files, we began developing scripts to process gene and metabolic network data for the BioCyc database. BioCyc is a public, curated database of biological pathway data. Since it contains both stoichiometric information and the corresponding gene regulatory network, constructing our netlist from this database would be very useful and help give our simulation tool broader applicability.

We divide the metabolites into three categories reactants, products, and reactants/products. The reactants are metabolites that are inputs to the system and consumed only; these are set to initial voltage values of IC=1V. Products are metabolites that are made during the course of the simulation and never used to make other metabolites. Products are set to initial capacitance values of IC=$1e^{-5}$ while metabolites that are both reactants and products are set to initial IC=$1e^{-3}$. For this simulation we use a default reaction rate of 0.001 and use a global enzyme to regulate the metabolic reactions for the system.

Figure 7.2 shows the XDamp screenshot for a thirty minute Xyce simulation of *E. coli* whole-cell metabolic, circuit simulation. The horizontal axis is simulation time (in seconds) and the vertical is the voltage of each node, which corresponds to the concentration of each metabolite.

7.2 The Hybrid Whole-Cell Model

We construct a hybrid whole-cell model of $E.coli$ that incorporates stoichiometric and gene regulatory data into one circuit. Due to the availability and accessibility of data, we focus on the central metabolic network and the genes that regulate central metabolism in $E.coli$. The central metabolism metabolic and gene network model is based on supplemental data from Covert and Palsson 2002[22]. The Covert-Palsson model (Figure 7.3 contained sixteen regulatory proteins with forty-five of the seventy-three enzymes regulated.

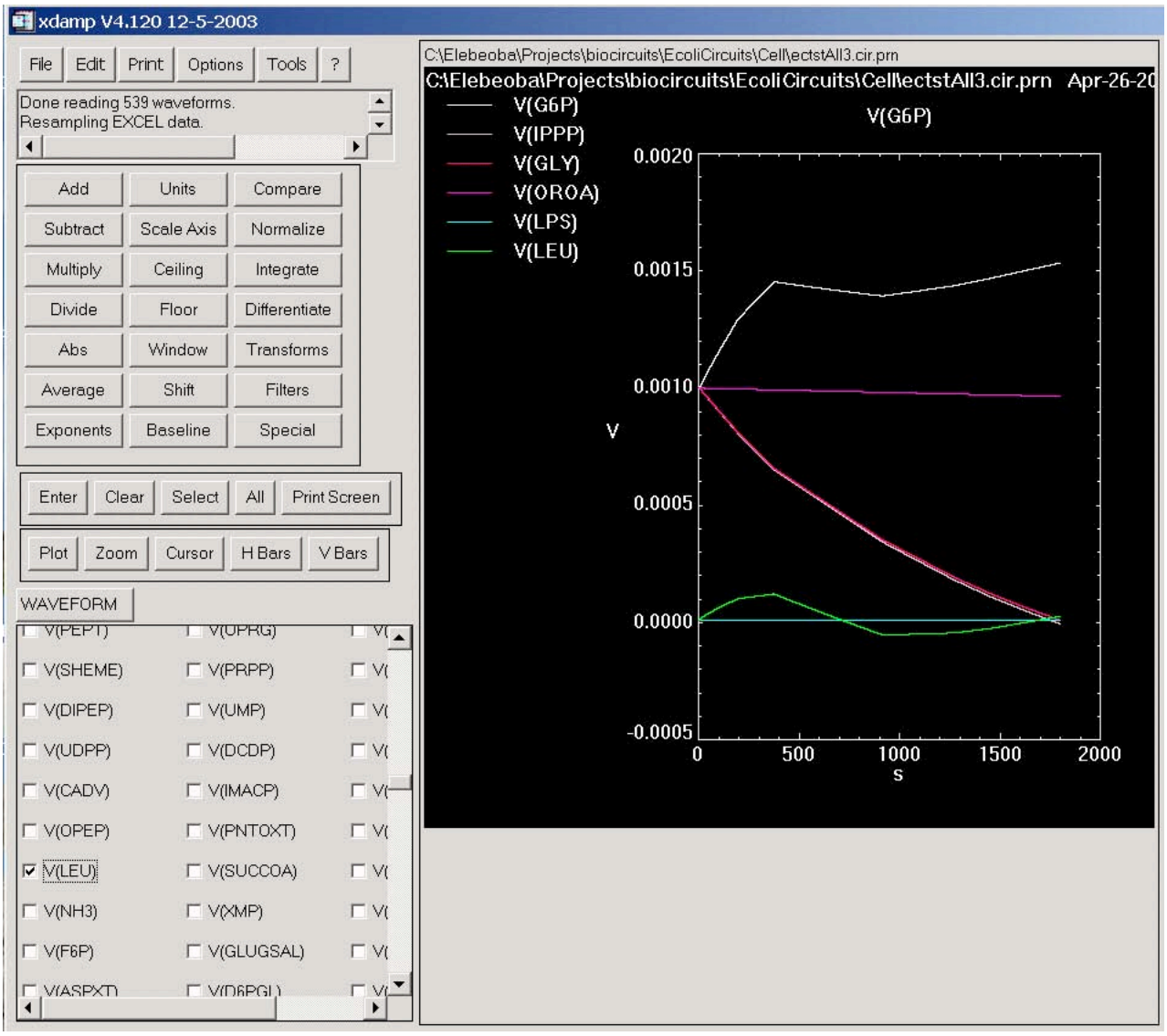

Figure 7.2. XDamp screenshot for the Xyce simulation results of *E. coli* whole-cell, circuit simulation.

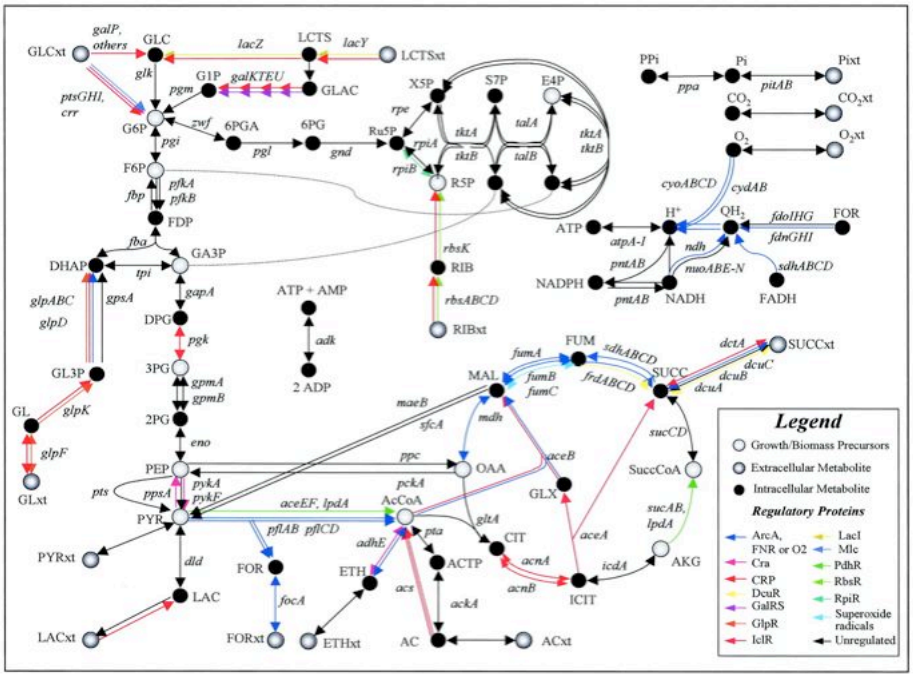

Figure 7.3. The Covert-Palsson model of $E.coli$ central metabolism.

Coupling Metabolic and Genetic Components

Metabolic components of the netlist are generated using $path2cir.pl$. The genetic regulatory components were then incorporated manually and the corresponding multi-input logic subcircuits were included. The metabolic and genetic components are coupled using the transmission gate component previously described. As an example, to implement the metabolic reaction mediated by isocitrate lyase, we implement the metabolic reaction

$$ICIT \rightarrow GLX + SUCC$$

using the genetic regulatory circuit

$$aceA = NOT(lclR)$$

we use the following netlist:

```
*Metabolic reaction controlled by AceA gene
** ACEA :  ( Isocitrate lyase  )   ICIT > GLX + SUCC
xRxn_ICIT_GLXSUCC ICIT GLX SUCC AceA AceANot nVdd nVddN Rxn1To2
   PARAMS: r1Stio=1 p1Stio=1 p2Stio=1 fRate=0.001

*Regulation of AceA
*aceA = IF (not (IclR))
xGn_aceA IclR AceA not
R_AceA AceA 0 100K
xGn_AceANot AceA AceANot not
R_AceANot AceANot 0 100K

*Subcircuits used for metabolic and gene network
*One to two metabolic subcircuit
**** Rxn1To2 ****
.SUBCKT Rxn1To2 rnt1 prd1 prd2 enzyme enzymeNot nVdd nVddN
   PARAMS: r1Stio=1 p1Stio=1 p2Stio=1 fRate=1.67e-5
.PARAM CapReact1=1
*** Enzyme activity as a Tx switch
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
*** Reaction - consume the reactants
BReact1 rnt1 0 I={r1Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) ) }
*** Reaction - produce the products
BProd1 0 prd1 I={p1Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) ) }
BProd2 0 prd2 I={p2Stio * fRate * ( (CapReact1 * V(rnt1in))**(r1Stio) ) }
.ENDS

*Implementation of NOT subcircuit
***** NOT Subcircuit
```

```
.SUBCKT NOT in1 outF PARAMS: koutF=0.5
.PARAM vMax=1
.PARAM TNot=0.5
BoutF outF 0 V={SDT(U(vMax-V(in1) - TNot) - V(outF)*koutF)}
.ENDS
```

We have two reactant elements, external glucose, GLCxt, and external ribose, RIBxt. To simulate the growth of $E.coli$ on a glucose medium we set the initial value of GLCxt=10.4V and RIBxt=1V. $Biomass$, which is the cell biomass, is set to a negligible amount. All other reactant-product metabolites were set to an initial value of IC=1e-2 except for Acetate, which was set to IC=0.3V to be consistent with the initial conditions in Covert and Palsson 2002 [22]. We ran the netlist for a simulation time of ten hours. Figure 7.4 and Figure 7.5shows the results of the simulation for growth on a glucose medium for a thirty minute simulation. The simulation shows an increase in biomass as glucose is consumed,

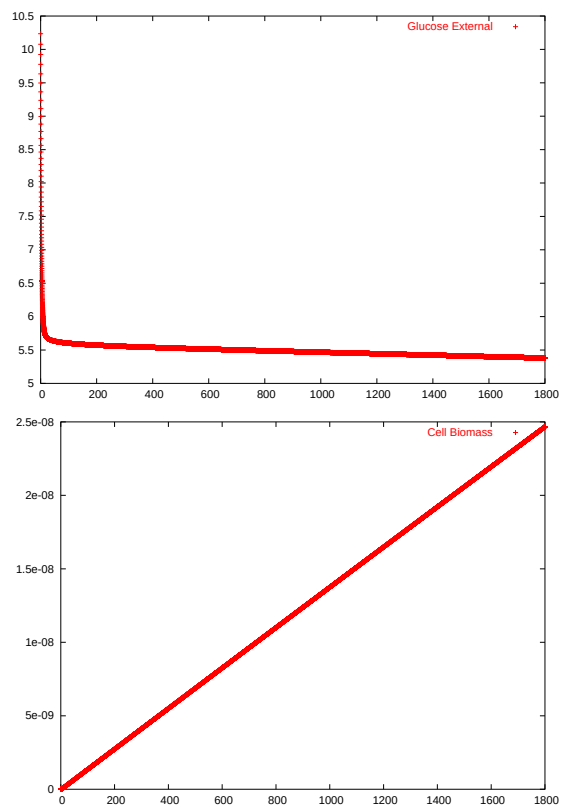

Figure 7.4. Xyce simulation (30 minutes) of hybrid $E.coli$ netlist. (Top) External glucose consumption and ($Bottom$) Cell biomass production.

which is the expected qualitative behavior.

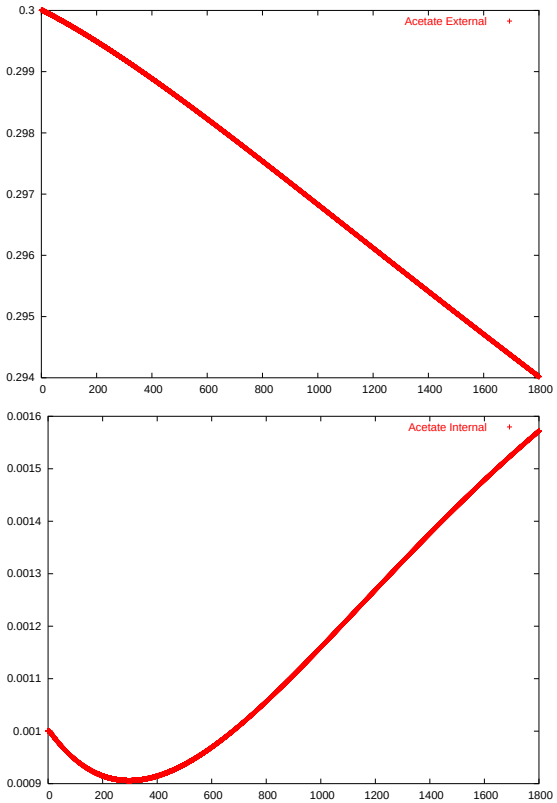

Figure 7.5. Xyce simulation (30 minutes) of hybrid $E.coli$ netlist. (Top) External acetate diffuses into cell and ($Bottom$) internal acetate increases.

The current implementation uses default reaction rates of 0.001. In order to incorporate empirically determined reaction rates, such as those available in the $BRENDA$ database, we modify the $BSOURCE$ implementation for simulating metabolic processes. We revisit the tryptophan biosynthesis pathway to demonstrate how we can incorporate empirical rate information into the Xyce simulation.

8. Tryptophan Revisited: Reformulating the Metabolic Subcircuit

A challenge in the development of accurate biological network simulations for systems biology is the availability of accurate rate data for metabolic reactions. Databases like $BRENDA$ (*http://www.brenda.uni-koeln.de/*)contain empirically determined reaction rates for various organisms including $E.coli$. Incorporation of this data will reduce the number of unknown simulation parameters and provide boundaries for model refinement. Revisiting our earlier Xyce model of tryptophan biosynthesis, we reformulate the $BSOURCE$ model to incorporate specific rate variables, incorporate rates from $BRENDA$, and couple the Xyce model to the DAKOTA optimization framework to refine remaining model parameters based on experimental time series data.

8.1 Reformulation of the Metabolic BSOURCE Element

It is reasonable to formulate the metabolic reactions in a format that lends itself to available data. We use the Michaelis-Menten equation to describe the reaction kinetics using the Michaelis-Menten constant, K_M, and the turnover rate, K_{cat}. Both rates are reported in the $BRENDA$ database.

The Michaelis-Menten equation for an enzyme catalyzed reaction provides an algebraic description of the formation of a product P from substrate S, which is modified by the enzyme E [23].

$$\text{E + S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \text{ES} \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \text{E + P} \tag{8.1}$$

As Equation 8.1 shows, The substrate must first associate with the enzyme, forming the enzyme-substrate complex, ES at an association rate of k_1. ES disassociates into enzyme and product at a rate of k_2 and disassociates into enzyme and unmodified substrate at a rate of k_{-1}. The reaction rate constants are k_1, k_{-1}, and k_2. Early in the reaction the concentration of product $[P]$ is negligible and we make the simplifying assumption that the reverse resaction, $P \rightarrow ES$, can be ignored. The reaction rate, V, is limited by the second step, conversion of ES to P and can be written as

$$V = K_2[ES]$$

where $[ES]$ is the concentration of the enzyme-substrate complex. Assuming an [S] $\gg$ [E] and that the system is in steady-state (relative equality in the formation and breakdown of the enzyme-substrate complex) the following algebraic relationship emerges:

$$\frac{k_1[E][S]}{k_{-1}+k_2} = [ES] \tag{8.2}$$

The algebraic form for the reaction rate can be written as

$$V = \frac{dP}{dt} = \frac{k_2[E_t][S]}{\frac{k_{-1}+k_2}{k_1} + [A]} \tag{8.3}$$

where E_t represents the total enzyme concentration. For now if we assume this is constant for all reactions we can further simplify the reaction rate equation. The turnover rate, K_{cat}, is equivalent to the number of substrate molecules converted to product in a given unit of time on a single enzyme molecule when the enzyme is saturated with substrate [23]. Turnover rate can be equivalenced to the rate constant for the rate-limiting step. Assuming that the breakdown of the enzyme-substrate complex to product, $[ES] \rightarrow [P]$, is the rate limiting step then we can equate turnover rate to k_2. For the single substrate/single product reaction, the reaction rate is

$$V = \frac{dP}{dt} = \frac{K_{cat}[S]}{K_M + [A]} \tag{8.4}$$

where

$$K_M = \frac{k_{-1}+k_2}{k_1} \tag{8.5}$$

We can extend this analysis to multiple substrate/multiple product systems if we maintain the steady state assumption, constant total enzyme concentration, and that the breakdown of the enzyme-substrates complex is the limiting step. These assumptions cause the rate equation to depend mainly on the number of inputs. For the tryptophan biosynthesis metabolic network, the rate equations are of the form shown in Table 8.1 (derived using the King-Altman method as described in *http://www.biokin.com/king-altman/theory.html*):

Below is an excerpt from the modified netlist for tryptophan biosynthesis that shows the modified metabolic circuit and corresponding subcircuits, which simulates metabolic reactions using the previously presented form.

```
*************************************************************************
Tryptophan Biosynthesis Circuit

.INCLUDE "param.cir"

** chor + gln -> glu + pyr + an
xRxn_ChorGln_GluPyrAn chor gln glu pyr an trpEDCBA trpEDCBANot
nVdd nVddN Rxn2To3 PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1
p3Stio=1 km1={km_chor} km2={km_gln} kcat={kcat_ChorGln}
```

Table 8.1. Multi-substrate/multi-product reaction rate formulation.

Reaction	Rate Equation	K_M	K_{cat}
$\text{E + S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \text{ES} \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \text{E + P}$	$\frac{K_{cat}[S]}{K_M+[P]}$	$\frac{k_{-1}+k_2}{k_1}$	k_2
A+B → P+Q:	$\frac{K_{cat}[A][B]}{[A][B]+K_{M,B}[A]+K_{M,A}K_{M,B}}$	$K_{M,A} = \frac{k_{-1}+k_3}{k_1}$ $K_{M,B} = \frac{k_{-2}+k_3}{k_2}$	k_3
$\text{A + E} \overset{k_1}{\rightarrow} \text{EA}$			
$\text{EA + B} \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \text{EAB}$			
$\text{EAB} \overset{k_3}{\rightarrow} \text{P + EQ}$			
$\text{EQ} \overset{k_4}{\rightarrow} \text{Q + E}$			

```
** an + prpp -> ppi + npran
xRxn_AnPrpp_PpiNpran an prpp ppi npran trpEDCBA trpEDCBANot
nVdd nVddN Rxn2To2 PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1
km1={km_an} km2={km_prpp} kcat={kcat_AnPrpp}

** npran -> cpad5p
xRxn_Npran_Cpad5p npran cpad5p trpEDCBA trpEDCBANot nVdd nVddN
Rxn1To1  PARAMS: r1Stio=1 p1Stio=1  km1={km_npran} kcat={kcat_Npran}

** cpad5p -> co2 + igp
xRxn_Cpad5p_Co2Igp cpad5p co2 igp trpEDCBA trpEDCBANot nVdd nVddN
Rxn1To2  PARAMS: r1Stio=1 p1Stio=1 p2Stio=1  km1={km_cpad5p}
kcat={kcat_Cpad5p}

** igp + ser -> t3p1 + trp
xRxn_IgpSer_T3p1Trp igp ser t3p1 trp trpEDCBA trpEDCBANot nVdd nVddN
Rxn2To2  PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1  km1={km_igp}
km2={km_ser} kcat={kcat_IgpSer}

** Analysis
.TRAN 10 300 0 1

************************* Reaction Subcircuits
**** Rxn1To1 ****
.SUBCKT Rxn1To1 rnt1 prd1 enzyme enzymeNot nVdd nVddN
```

```
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 p3Stio=1  km1=1 kcat=1
*
* Modeling the simple reaction set of:
*      r1Stio A -> p1Stio B
*
* with rate constant derived using King-Altman method, s.t.
* v = kcat [A] / (km1 + [A])
*
*** Enzyme activity as a Tx switch
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
*
***Reaction - consume the reactants
**EMay mod to use URAMP function to prevent neg substrate vals
BCons1 rnt1 0 I={(1/r1Stio) *((kcat*URAMP(V(rnt1in)))/(km1 + URAMP(V(rnt1in))))}

***Reaction - produce products
BProd1 0 prd1 I={(1/p1Stio) *((kcat*URAMP(V(rnt1in)))/(km1 + URAMP(V(rnt1in))))}
*
.ENDS


**** Rxn1To2 ****
.SUBCKT Rxn1To2 rnt1 prd1 prd2 enzyme enzymeNot nVdd nVddN
PARAMS: r1Stio=1 p1Stio=1 p2Stio=1 km1=1 kcat=1
*
* Modeling the simple reaction set of:
*      r1Stio A -> p1Stio B1 + p2Stio B2
*
* with rate constant derived using King-Altman method, s.t.
* v = kcat [A] / (km1 + [A])
*
*** Enzyme activity as a Tx switch
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
*
***Reaction - consume the reactants
BCons1 rnt1 0 I={(1/r1Stio) *((kcat*URAMP(V(rnt1in)))/(km1 + URAMP(V(rnt1in))))}

***Reaction - produce products
BProd1 0 prd1 I={(1/p1Stio) *((kcat*URAMP(V(rnt1in)))/(km1 + URAMP(V(rnt1in))))}
BProd2 0 prd2 I={(1/p2Stio) *((kcat*URAMP(V(rnt1in)))/(km1 + URAMP(V(rnt1in))))}
*
*
.ENDS
```

```
**** Rxn2To2 ****
.SUBCKT Rxn2To2 rnt1 rnt2 prd1 prd2 enzyme enzymeNot nVdd nVddN
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 km1=1 km2=1 kcat=1
*
* Modeling the simple reaction set of:
*      r1Stio A1 + r2Stio A2  -> p1Stio B1 + p2Stio B2
*
* with rate constant derived using King-Altman method, s.t.
* v = kcat [A1][A2] / ([A1][A2] + km2[A1] + km1km2)
*
*** Enzyme activity as a Tx switch
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
xtx_rnt2 rnt2 rnt2in enzymeNot enzyme nVdd nVddN tx
Rres_rnt2in rnt2in 0 100K
*
***Reaction - consume the reactants
BCons1 rnt1 0 I={(1/r1Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
BCons2 rnt2 0 I={(1/r2Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}

***Reaction - produce products
BProd1 0 prd1 I={(1/p1Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
BProd2 0 prd2 I={(1/p2Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
*
.ENDS


**** Rxn2To3 ****
.SUBCKT Rxn2To3 rnt1 rnt2 prd1 prd2 prd3 enzyme enzymeNot nVdd nVddN
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 p3Stio=1  km1=1 km2=1 kcat=1
**.SUBCKT Rxn2To3 rnt1 rnt2 prd1 prd2 prd3 enzyme enzymeNot nVdd nVddN
PARAMS: r1Stio=1 r2Stio=1 p1Stio=1 p2Stio=1 p3Stio=1  km1=1 km2=1 kcat=1
*
* Modeling the simple reaction set of:
*      r1Stio A1 + r2Stio A2  -> p1Stio B1 + p2Stio B2 + p3Stio B3
*
* with rate constant derived using King-Altman method, s.t.
* v = kcat [A1][A2] / ([A1][A2] + km2[A1] + km1km2)
*
*** Enzyme activity as a Tx switch
```

```
xtx_rnt1 rnt1 rnt1in enzymeNot enzyme nVdd nVddN tx
Rres_rnt1in rnt1in 0 100K
xtx_rnt2 rnt2 rnt2in enzymeNot enzyme nVdd nVddN tx
Rres_rnt2in rnt2in 0 100K
*
***Reaction - consume the reactants
BCons1 rnt1 0 I={(1/r1Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
BCons2 rnt2 0 I={(1/r2Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}

***Reaction - produce products
BProd1 0 prd1 I={(1/p1Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
BProd2 0 prd2 I={(1/p2Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
BProd3 0 prd3 I={(1/p3Stio) *((kcat*URAMP(V(rnt1in))*URAMP(V(rnt2in)))/
(URAMP(V(rnt1in))*URAMP(V(rnt2in)) + km2*URAMP(V(rnt1in)) + km1*km2))}
*
.ENDS
*****************************************************************************
```

8.2 Optimization Framework for Model Refinement

Using the rate reaction format in Table 8.1, we simulate the tryptophan synthesis, incorporating empirically determined rate information from the $BRENDA$ database through a parameter file. Below is a list of the fixed parameters used in the simulation.

```
.PARAM km_chor=1.2000000000e-03
.PARAM km_gln=3.6000000000e-01
.PARAM km_prpp=1.0000000000e-01
.PARAM km_npran=4.9000000000e-03
.PARAM km_cpad5p=3.4000000000e-04
.PARAM kcat_AnPrpp=4.4000000000e+00
.PARAM kcat_Npran=5.0000000000e+01
.PARAM kcat_Cpad5p=2.2000000000e+00
```

As in the Drosophila tissue differentiation simulations, we couple the Xyce netlist for tryptophan biosynthesis to $DAKOTA$ optimization environment to find the optimal rate constants for five of the thirteen rate constants in the metabolic pathway:

$$K_M^{AN}, K_M^{IGP}, K_M^{SER}, K_{cat}^{ChorGln}, K_{cat}^{IgpSer}$$

Using empirical data from Bliss 1979 [24] we performed a least squares analysis, comparing the output of $Xyce$ (300 seconds of simulation data) to the data from Bliss. The Perl script below was used to process the $Xyce$ .PRN file and perform an error analysis based on the empirical data:

```
***********************************************************************
#!/usr/local/bin/perl -w
#
# cmprSim2ExpData.pl <xyce output file>
#Compares Xyce simulation output files to experimental data for the
# simulated system.
#

# Begin Main package space
{
my $index;
my $indexMin;
my $indexMax;
my $simTime;
my $enzyme;
my $tryp;
my $line;
my @lineA;
my %trypExpData;
my %trypSimData;
my $maxExpTime=300;
my $timeKey;
my $objectiveF;

# Load experimental data for each available time point;
$trypExpData{0}=35;
$trypExpData{120}=100;
$trypExpData{180}=150;
$trypExpData{240}=200;
$trypExpData{300}=250;

# Initialize simulated data hash
$trypSimData{0}{sum}=0;
$trypSimData{0}{npnts}=1;
$trypSimData{120}{sum}=0;
$trypSimData{120}{npnts}=1;
$trypSimData{180}{sum}=0;
$trypSimData{180}{npnts}=1;
$trypSimData{240}{sum}=0;
$trypSimData{240}{npnts}=1;
```

```
$trypSimData{300}{sum}=0;
$trypSimData{300}{npnts}=1;


#read in info
$line=<>; #read in first line
$simTime=0;
while($line=<>)
{
if ($simTime < $maxExpTime)
{
($index, $simTime, $enzyme, $tryp, @lineA)=split ' ',$line;
#remove excess whitespace
$index=~s/\s//g;
if ($index ne 'End')
{
$simTime=~s/\s//g;
$simTime=int($simTime);
if ($trypSimData{$simTime}{npnts})
{
#add to sim data hash
$tryp=~s/\s//g;
$trypSimData{$simTime}{sum} = $trypSimData{$simTime}{sum} + $tryp;
$trypSimData{$simTime}{npnts} = $trypSimData{$simTime}{npnts} + 1;
##print "\n --$index--$simTime--$enzyme--$tryp-- \n";
}
} #end if index ne end
} # end if simtime lt maxexptime
} # end while
# calculate objective function and print it out
$objectiveF=0;
foreach $timeKey (keys %trypExpData)
{
$trypSimData{$timeKey}{npnts}=$trypSimData{$timeKey}{npnts}-1;
if ($trypSimData{$timeKey}{npnts} > 0)
{
$objectiveF=$objectiveF + ($trypExpData{$timeKey} -
($trypSimData{$timeKey}{sum}/$trypSimData{$timeKey}{npnts}));
$trypSimData{$timeKey}{objF}=$trypExpData{$timeKey} -
($trypSimData{$timeKey}{sum}/$trypSimData{$timeKey}{npnts});
}
}
# for least squares
printf "%.20f f \n", $trypSimData{0}{objF};
printf "%.20f f \n", $trypSimData{120}{objF};
```

```
printf "%.20f f \n", $trypSimData{180}{objF};
printf "%.20f f \n", $trypSimData{240}{objF};
printf "%.20f f \n", $trypSimData{300}{objF};

}
# end main package space
exit;
*************************************************************************
```

Based on the error analysis, $DAKOTA$ generated new values for the rate constants, which were incorporated into a parameter file included in the $Xyce$ netlist. Figure 8.1 shows the results of several $DAKOTA$-$Xyce$ iterations. The horizontal axis is time and the vertical

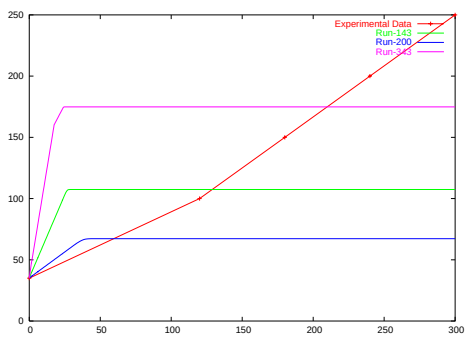

Figure 8.1. Comparison of Xyce tryptophan biosynthesis to experimental data.

shows the concentration of tryptophan for the empirical data set and the simulated data set. Plots of externally provided and intermediate metabolites show quick consumption of substrate metabolites. Optimization over all rate constants and initial values for externally provided substrates may result in successful parameter estimation for the given empirical data.

Coupling the $Xyce$ biological circuit simulation tool with $DAKOTA$ and available empirical data provides a framework for elucidating unknown reaction rates and provides a more reliable models for systems biology applications. The approach used in this chapter can be extended to refine whole-cell network models and provide quantitative results that will be useful to experimentalists.

Biochemical Circuit Devices

This appendix includes circuit devices useful to biological and chemical simulations. The devices are presented as subcircuits as this make the underlying operations evident and allows one to change the devices if needed.

A Chemcial channel

```
*
* Chemical channnel subcircuit
*
* This is simple a capacitor to allow for the accumulation
* of material and a path to ground
*
.SUBCKT ChemCh chNode PARAMS: chCon=1e-16

Cch1 chNode 0 1   IC=chCon
Rrch1 chNode 2 1e12

.ENDS
```

This page is left intentionally blank

B Power law based reactions

```
*
* Simple Reactions and Analogs as Subcircuit
*
* For each of the following reactions, we
* assume that there is a 1 farad capacitor on
* each node entering each subcurcuit.  While 1 farad
* capacitors are unrealistic, they make
* interpretation of the results easier, i.e. voltage
* on a given capicator will equal the molar
* concentration of given species.  If we included a
* capacitor on each subcurcuit, then we would end up
* with multiple capacitors in parallel and that would
* confuse the interpretation of restuls


*
*
* Reactions starting with 1 reactant ---------------------------
*
*

.SUBCKT Rxn1To1 rnt1 prd1
+ PARAMS: r1Stio=1
+         p1Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A -> p1Stio B
*
* with a forward rate constant of fRate.
*

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * V(rnt1)**(r1Stio)
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * V(rnt1)**(r1Stio)
*
.ENDS
```

```
.SUBCKT Rxn1To2 rnt1 prd1 prd2
+ PARAMS: r1Stio=1
+         p1Stio=1 p2Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A -> p1Stio B + p2Stio C
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
*
.ENDS


.SUBCKT Rxn1To3 rnt1 prd1 prd2 prd3
+ PARAMS: r1Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A -> p1Stio B + p2Stio C + p3Stio D
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( SGN( V(rnt1) ) *
                                  ( V(rnt1)**(r1Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( SGN( V(rnt1) ) *
```

```
+                                  ( V(rnt1)**(r1Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
*
.ENDS


.SUBCKT Rxn1To4 rnt1 prd1 prd2 prd3 prd4
+  PARAMS: r1Stio=1
+          p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1
+          fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A -> p1Stio B + p2Stio C + p3Stio D + p4Stio E
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( SGN( V(rnt1) ) *
+                                  ( V(rnt1)**(r1Stio)) )
*
.ENDS


.SUBCKT Rxn1To5 rnt1 prd1 prd2 prd3 prd4 prd5
+ PARAMS: r1Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1 p5Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A -> p1Stio B + p2Stio C + p3Stio D + p4Stio E
*
* with a forward rate constant of fRate.
```

```
* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
BProd5 0 prd5 I=p5Stio * fRate * ( SGN( V(rnt1) ) *
+                                 ( V(rnt1)**(r1Stio)) )
*
.ENDS




*
*
* Reactions starting with 2 reactants ---------------------------
*
*


.SUBCKT Rxn2To1 rnt1 rnt2 prd1
+ PARAMS: r1Stio=1 r2Stio=1
+         p1Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B -> p1Stio C
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
```

```
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*
.ENDS


.SUBCKT Rxn2To2 rnt1 rnt2 prd1 prd2
+ PARAMS: r1Stio=1 r2Stio=1
+         p1Stio=1 p2Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B -> p1Stio C + p2Stio D
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*
.ENDS


.SUBCKT Rxn2To3 rnt1 rnt2 prd1 prd2 prd3
+ PARAMS: r1Stio=1 r2Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B -> p2Stio C + p2Stio D
*
* with a forward rate constant of fRate.
```

```
* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*
.ENDS


.SUBCKT Rxn2To4 rnt1 rnt2 prd1 prd2 prd3 prd4
+ PARAMS: r1Stio=1 r2Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B ->
*        p2Stio C + p2Stio D + p3Stio E + p4Stio F
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
```

```
*
.ENDS


.SUBCKT Rxn2To5 rnt1 rnt2 prd1 prd2 prd3 prd4 prd5
+ PARAMS: r1Stio=1 r2Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1 p5Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B ->
*        p2Stio C + p2Stio D + p3Stio E + p4Stio F + p5Sto G
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
BProd5 0 prd5 I=p5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                                  ( V(rnt2)**(r2Stio)) )
*
.ENDS



*
*
* Reactions starting with 3 reactants --------------------------
*
*
```

```
.SUBCKT Rxn3To1 rnt1 rnt2 rnt3 prd1
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1
+         p1Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C -> p1Stio D
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*
.ENDS


.SUBCKT Rxn3To2 rnt1 rnt2 rnt3 prd1 prd2
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1
+         p1Stio=1 p2Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C -> p1Stio D + p2Stio E
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
```

```
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*
.ENDS


.SUBCKT Rxn3To3 rnt1 rnt2 rnt3 prd1 prd2 prd3
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C ->
*        p1Stio D + p2Stio E + p3Stio F
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*
.ENDS


.SUBCKT Rxn3To4 rnt1 rnt2 rnt3 prd1 prd2 prd3 prd4
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1
+         fRate=1
*
```

```
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C ->
*        p1Stio D + p2Stio E + p3Stio F + p4Stio G
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*
.ENDS


.SUBCKT Rxn3To5 rnt1 rnt2 rnt3 prd1 prd2 prd3 prd4 prd5
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1 p5Stio=1
+         fRate=1
*
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C ->
*       p1Stio D + p2Stio E + p3Stio F + p4Stio G + p5Stio H
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
```

```
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
BProd5 0 prd5 I=p5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+                ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) )
*
.ENDS




*
*
* Reactions starting with 4 reactants --------------------------
*
*


.SUBCKT Rxn4To1 rnt1 rnt2 rnt3 rnt4 prd1
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1
+         p1Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D -> p1Stio E
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
```

```
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


.SUBCKT Rxn4To2 rnt1 rnt2 rnt3 rnt4 prd1 prd2
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1
+         p1Stio=1 p2Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D -> p1Stio E + p2Stio F
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


.SUBCKT Rxn4To3 rnt1 rnt2 rnt3 rnt4 prd1 prd2 prd3
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1
+         fRate=1
```

```
* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D ->
*        p1Stio E + p2Stio F + p3Stio G
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


.SUBCKT Rxn4To4 rnt1 rnt2 rnt3 rnt4 prd1 prd2 prd3 prd4
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D ->
*       p1Stio E + p2Stio F + p3Stio G + p4Stio H
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
```

```
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


.SUBCKT Rxn4To5 rnt1 rnt2 rnt3 rnt4 prd1 prd2 prd3 prd4 prd5
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1 p5Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D ->
*        p1Stio E + p2Stio F + p3Stio G + p4Stio H + p5Stio I
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
```

```
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
BProd5 0 prd5 I=p5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


*
*
* Reactions starting with 5 reactants ---------------------------
*
*


.SUBCKT Rxn5To1 rnt1 rnt2 rnt3 rnt4 rnt5 prd1
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1 r5Stio=1
+         p1Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D + r5Stio E -> p1Stio F
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons5 rnt5 0 I=r5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
```

```
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS


.SUBCKT Rxn5To2 rnt1 rnt2 rnt3 rnt4 rnt5 prd1 prd2
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1 r5Stio=1
+         p1Stio=1 p2Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D + r5Stio E ->
*        p1Stio F + p2Stio G
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons5 rnt5 0 I=r5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*
```

```
.ENDS


.SUBCKT Rxn5To3 rnt1 rnt2 rnt3 rnt4 rnt5 prd1 prd2 prd3
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1 r5Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D + r5Stio E ->
*         p1Stio F + p2Stio G + p3Stio H
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons5 rnt5 0 I=r5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS
```

```
.SUBCKT Rxn5To4 rnt1 rnt2 rnt3 rnt4 rnt5 prd1 prd2 prd3 prd4
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1 r5Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D + r5Stio E ->
*           p1Stio F + p2Stio G + p3Stio H + p4Stio I
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons5 rnt5 0 I=r5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS
```

```
.SUBCKT Rxn5To5 rnt1 rnt2 rnt3 rnt4 rnt5 prd1 prd2 prd3 prd4 prd5
+ PARAMS: r1Stio=1 r2Stio=1 r3Stio=1 r4Stio=1 r5Stio=1
+         p1Stio=1 p2Stio=1 p3Stio=1 p4Stio=1 p5Stio=1
+         fRate=1

* In this subcircuit we're modeling the simple reaction
* set of:
*      r1Stio A + r2Stio B + r3Stio C + r4Stio D + r5Stio E ->
*         p1Stio F + p2Stio G + p3Stio H + p4Stio I + p5Stio J
*
* with a forward rate constant of fRate.

* Forward Reaction
*   consume the reactants
BCons1 rnt1 0 I=r1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons2 rnt2 0 I=r2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons3 rnt3 0 I=r3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons4 rnt4 0 I=r4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BCons5 rnt5 0 I=r5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*   produce a product
BProd1 0 prd1 I=p1Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd2 0 prd2 I=p2Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd3 0 prd3 I=p3Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd4 0 prd4 I=p4Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
BProd5 0 prd5 I=p5Stio * fRate * ( ( V(rnt1)**(r1Stio)) *
```

```
+ ( V(rnt2)**(r2Stio)) * ( V(rnt3)**(r3Stio)) *
+ ( V(rnt4)**(r4Stio)) * ( V(rnt4)**(r4Stio)) )
*
.ENDS
```

C Enzymaticly controlled reactions

```
*
* Simple Transformations of a species.
*
* Here we model B -> B' not as a reaction
* but as a maximally limited rate proces
* i.e. a reaction roughly linear at low
* concentration but limited at hight
* driving rate.
*
* Typically,
*
* dB/dt = vmax ( A**vrate / (ahalf + A**vrate) );
*
* Where A is not consumed (acts like an enzyme)
* but B and B' are changed
*
* Promoters and Repressors can influcence
* the maximal rate and are handled by additional
* subcircuits.
*

.FUNC enzRate( pc, pwr, phalf )  (pc)**(pwr) /
+                ( (phalf)**(pwr) + (pc)**(pwr) )

*
*
* A -> A' with one promoter
*
*
.SUBCKT Tfm1To1w1p rnt1 prd1 prom1
+ PARAMS: vmax=1
+         phalf=1
+         prate=1

*   remove reactants
BReac1 rnt1 0 I= V(rnt1) * vmax * ( V(prom1)**(prate) ) /
+                    (phalf**(prate) + V(prom1)**(prate))

*   produce a product
BProd1 0 prd1 I= V(rnt1) * vmax * ( V(prom1)**(prate) ) /
+                    (phalf**(prate) + V(prom1)**(prate))
```

```
.ENDS


*
*
* A -> A' with one promoter and one repressor
*
*
.SUBCKT Tfm1To1w1p1r rnt1 prd1 prom1 rep1
+ PARAMS: vmax=1
+         phalf=1
+         prate=1
+         rhalf=1
+         rrate=1

*   remove reactants
BReac1 rnt1 0 I= V(rnt1) * vmax *
+  ( ( (V(prom1)* (1.0 - enzRate( V(rep1), rrate, rhalf )))**(prate) ) /
+  ((phalf)**(prate) + (V(prom1)*
+   (1.0 - enzRate( V(rep1), rrate, rhalf )))**(prate)) )
*   produce a product
BProd1 0 prd1 I= V(rnt1) * vmax *
+  ( ( (V(prom1)* (1.0 - enzRate( V(rep1), rrate, rhalf )))**(prate) ) /
+  ((phalf)**(prate) + (V(prom1)*
+   (1.0 - enzRate( V(rep1), rrate, rhalf )))**(prate)) )

.ENDS
```

Bibliography

[1] C. Nussleinvolhard and E. Wieschaus. Mutations affecting segment number and polarity in drosophila. *Nature*, 287(5785):795–801, 1980.

[2] Harley H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–656, 1995.

[3] Harley H. McAdams and Arkin A. Gene regulation: Towards a circuit engineering discipline. *Current Biology*, 10:318–320, 2000.

[4] Harley H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annu. Rev. Biophys. Biomol. Struct.*, 27:199–224, 1998.

[5] A.P. Arkin. Synthetic cell biology. *Current Opinion in Biotechnology*, 12:638–644, 2001.

[6] B. Palsson. The challenges of in silico biology. *Nature Biotechnology*, 18:1147–1150, 2000.

[7] *Xyce: Parallel Electronic Simulator Reference Guide*. http://www.cs.sandia.gov/xyce, 2003.

[8] *Dakota: A Multilevel Parallel, Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification and Sensitivity Analysis*. http://software.sandia.gov, 2004.

[9] Paul Horowitz and Winfield Hill. *The Art of Electronics, 2nd ed.* Cambridge University Press, Cambridge, 1989.

[10] J.J Tyson. *Oscillations and Traveling Waves in Chemical Systems*. John Wiley & Sons, 1985.

[11] George von Dassow, E. Meir, E. Munro, and G. Odell. The segment polarity network is a robust development module. *Nature*, 406:188–192, 2000.

[12] Ghabrial Amin, S. Lusching, M. Metzstein, and M. Krasnow. Branching morphogenesis of the *Drosophila* tracheal system. *Annu. Rev. Cell Dev. Biol.*, 19:623–47, 2003.

[13] N. Ingolia. Topology and robustness in the drosophila segment polarity network. *PLOS Biology*, 2(6):805–815, 2004.

[14] Christophe H. Schilling, Jeremy S. Edwards, and David Letscher. Combining pathway analysis with flux balance analysis for the comprehensive study of metabolic systems. *Biotechnology and Bioengineering*, 71(4):286–306, 2000/2001.

[15] Markus W. Covert, Christophe H. Schilling, and Bernhard Palssson. Regulation of gene expression in flux balance models of metabolism. *Journal of Theoretical Biology*, 213:73–88, 2001.

[16] Paul Smolen, Douglas A. Baxter, and John H. Byrne. Modeling transcriptional control in gene networks - methods, recent results, and future directions. *Bulletin of Mathematical Biology*, 62:247–292, 2000.

[17] Zhi-Long Xiu, An-Ping Zeng, and Wolf-Dieter Deckwer. Model analysis concerning the effects of growth rate and intracellular tryptophan level on the stability and dynamics of tryptophan biosynthesis in bacteria. *Journal of Biotechnology*, 58:125–140, 1997.

[18] Necmettin Yildirim and Michael C. Mackey. Feedback regulation in the lactose operon: a mathematical modeling study and comparison with experimental data. *Biophysical Journal*, 84:2841–2851, May 2003.

[19] H. H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annual Review of Biophysics and Biomolecular Structure*, 27:199–224, 1998.

[20] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics*, 31:64–68, 2002.

[21] Shawn Martin, George Davidson, Elebeoba May, Margaret Werner-Washburne, and Jean-Loup Faulon. Inferring genetic networks from microarray data. In *ICSB*, 2004.

[22] Markus Covert and Bernhard Palsson. Transcriptional regulation in constraints-based metabolic models of escherichia coli. *Journal of Biological Chemistry*, 277(31):28058–28064, August 2002.

[23] David L. Nelson and Michael M. Cox. *Lehninger Principles of Biochemistry, 4th Edition*. W. H. Freeman and Company, New York, 2005.

[24] Richard D. Bliss. A specific method for determination of free tryptophan and endogenous tryptophan in *Escherichia coli*. *Analytical Biochemistry*, 93:390–398, 1979.