Modeling Toxin-Antitoxin System with the Gillespie Algorithm

Allan Abraham* and Guillaume Lambert†
Cornell University
School of Applied and Engineering Physics,
Cornell University, Ithaca, New York 14853
(Dated: August 12, 2019)

Persister cells are dormant cells that can survive inside of hosts for long periods of time when the majority of the cell population has died due to various stressors such as antibiotics, nutrient deficiency, high temperatures and more. The formation of these cells is relatively unknown, but it is theorized that the toxin-antitoxin system might be one of the mechanisms that cells utilize to change into this persistent state. The toxin-antitoxin system can stochastically fluctuate between toxic and normal states creating a strong bistable switch. In the toxic state, the growth of the cells is arrested which causes the cells to become persisters. In this paper, the Gillespie's stochastic simulation algorithm is used to model the bistability of this system. Two stable states are observed, indicating the impact that the toxin-antitoxin system has on the growth of a bacterial cell. Bacteria use the toxin-antitoxin system to create a bet-hedging subpopulation even during periods of low stress.

I. INTRODUCTION

Persister cells were first documented in 1944 by Joseph Bigger who was a medical doctor from the University of Dublin[1]. He treated a culture of *Staphylococcus* with penicillin. He noticed that, while most of the culture was lysed, a small subpopulation still formed which he dubbed "persisters." He theorized that these were dormant, non-dividing cells. In 1980, Harris Moyed treated a culture of *Escherichia coli* with ampicillin then let the surviving cells grow without the presence of the antibiotic[2]. He identified two remaining populations which were antibioticic-resistant mutants that grew in the presence of ampicillin and persister cells that did not grow in the presence of ampicillin.

Persister cells typically make up about 10^{-5} to 10^{-6} of the population[3]. This small subpopulation survives through a treatment of antibiotics while regular cells die. They survive because antibiotics require active targets while persister cells are dormant and nondividing[4]. This is known as persister tolerance. This state is not caused by genetic mutation[3]. This seems to be a common adaptive strategy for cell survival as this allows cells to live through therapeutic and/or environmental stressors[5]. After these stressors are removed, the cells returns to its regular active state and continues growing. Persisters are responsible for 65% to 80% of bacterial infections[6]. Persisters are the result of both stochastic and deterministic events[4]. These events are not well understood but one proposed way is through toxin-antitoxin (TA) systems[4].

The TA system is a maintenance mechanism[7]. This system produces and regulates toxin and antitoxin proteins. The toxin protein is stable and inhibits cellular growth. The antitoxin protein is degrades quickly and forms an inactive neutralized complex with the toxin[4]. An analysis of the transcriptome of persister cells have shown an increase in the expression of TA genes[5]. There are five classes of TAS. In this paper, I focus on a general model of the type II TAS as it is the most studied class. This system is particularly found in pathological bacteria therefore a model that elucidates the formation of persister cells could be the key in preventing chronic infections[8]. In this system, the operons for antitoxins are located after the operons for toxins[9]. The toxin represses the production of antitoxins and its own. Antitoxins bind to toxins to form a neutralized pair. In this paper, I use the Gillespie algorithm to simulate the Type II toxin-antitoxin system.

Since the formation of persisters and the TAS fluctuate stochastically, the Gillespie's stochastic simulation algorithm is a useful method to model these systems[10]. This algorithm was developed by Joseph L. Doob in 1945 and presented by Dan Gillespie in 1976. The algorithm can show how important stochastic fluctuation, or noise, has on a system state. Simulating these fluctuations is extremely important for this model as the noise can radically switch the system from a normal state to a toxic, persistence, state and back to a normal state which forms a bistable switch. The algorithm can model this noise by utilizing random number generators. This algorithm is, also, event-driven. Events occur randomly at random times. At each time step, exactly one event occurs. This way, the algorithm can keep track of the changes in concentration of each molecule. This method is perfect for simulating the TA system as both, the TA system and the algorithm, are discrete and stochastic processes[10]. I use the Gillespie algorithm, in Python, to form a bistable switch from a general Type II toxin-antitoxin system and show how a small concentration of toxin proteins can have a cascading effect on the cell system.

* aa2655@cornell.edu
† lambert@cornell.edu

II. METHOD

The Gillespie algorithm consists of four steps: initialization, Monte Carlo step, update, and iterate[10].

a. Initialization Initial concentrations of toxin, antitoxin, and neutralized pairs are set and their values are saved to their respective arrays at `time = 0`. The total run time of the model is loaded and various constants are decided in this section.

b. Monte Carlo step The algorithm calculates the propensities of each state change in this section. There are five state changes that describe the dynamics of a generalized Type II toxin-antitoxin system. These state changes are described by the following equations:

$$w_1 = \frac{\beta_T}{1 + (\frac{T}{T_{half}})^n} \tag{1}$$

$$w_2 = \frac{\beta_A}{1 + (\frac{A}{A_{half}})^n} \tag{2}$$

$$w_3 = kAT \tag{3}$$

$$w_4 = \alpha_T T \tag{4}$$

$$w_5 = \alpha_A A \tag{5}$$

These equations were inspired by a paper written by Fasani et al and published in PNAS[11]. T and A represent the concentrations of free toxin and antitoxin proteins, respectively. Equations **1** and **2** are Hill equations that describe the rate of production of the toxin and antitoxin protein respectively. Toxin concentration is in the denominator of both equations because toxins repress the production of antitoxins and itself. The numerator of β is a scaling factor that can be used to describe ribosome binding site affinity and a burst production factor respectively. This scaling factor can be used to speed up or slow down the production either of the two proteins. T_{half} and A_{half} are the half-maximal concentration constants for toxins and antitoxins. n is the Hill number. Equation **3** describes the rate of the neutralized dimer pair formation which depends on a constant `k` and the concentrations of toxins and antitoxins. This equation is Equations **4** and **5** describe the rate of degradation of both proteins which depends on a constant α and the concentrations of toxin and antitoxin respectively for each equation. Propensity, the probability of a reaction r occurring, is described by the following equation[10]:

$$P_r = \frac{w_r}{\sum_{i=1}^{R} w_i} \tag{6}$$

R, in this equation, is the total number reactions which is five in this model.

c. Update Two random numbers, x_1 and x_2 are generated through the following line of code: `np.random.rand()` from the numpy library. This command generates a random number from a uniform distribution from 0 to 1. These two numbers are used to determine the time step to a reaction and which event occurs. Reaction r occurs when:[10]

$$P_{r-1} < x_1 \leq P_{r-1} + P_r \tag{7}$$

The state changes by drawing a sample from a discrete distribution. The probability that an event occurs is proportional to its propensity.[10] The time step is determined by a Poisson process that is computed stochastically by the reaction rates. This step is described in the following equation:

$$\tau = \frac{1}{\sum_{i=1}^{R} w_i} ln \frac{1}{x_2} \tag{8}$$

The concentrations of toxin, antitoxin, and neutralized pair are updated by which event occurs. The toxin or antitoxin concentrations increase by one if production occurs. The protein concentrations decreases by one if degradation occurs. If a neutralized pair formation event occurs, both concentrations of toxin and antitoxin are decreased by one. Time is updated by `T = t +` τ.

d. Iterate The program repeats the Monte Carlo and Update step till the reaction time reaches the simulation time. The program code, in its entirety, will be attached at the end of this paper.

III. RESULTS AND ANALYSIS

For the first run, I set the properties of toxin and antitoxin proteins to be the same to observe what stochastic fluctuations occurred. Fig. 1 shows that, in a symmetrical system, the system behaves as a bistable switch. Toxins dominate the first 4,000 minutes while there is a small amount of antitoxins produced. Around 4,000 minutes, the system randomly flips states, and antitoxins become the dominate population. During this time period, the cell would be in a normal growing state. At around 8,000 minutes, the system stochastically changes to the slow growing, persister state with toxin population being more prevalent without the balance of antitoxins. It is interesting to note that, if the toxin and antitoxin proteins had the same half-maximal concentration and degradation constant, then the system would be this unstable. Unfortunately, these values do not reflect reality as the antitoxin protein is more unstable than the toxin protein[7]. However, this run does confirm that the equations, that I use in conjunction with the Gillespie Algorithm, has the potential to model a bistable switch.

After testing this, I began using more realistic values from the paper by Fasani et al[11]. The intial values, of the runs that I will be presenting in this paper, can be

TABLE I: Initial parameters for each run. In every run: n = 2, simulation time = 10,080 (7 days), k = 0.0005.

Parameter	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6
T (nM)	0.0	0.0	0.0	100.0	0.0	0.0
A (nM)	0.0	0.0	0.0	0.0	0.0	0.0
A_{half} (nM)	40	40	40	40	60	60
T_{half} (nM)	40	40	40	40	40	100
β_{tox}	1	1	1	1	1	1
β_{Ant}	1	1	2	2	2	2
α_{tox}	0.0005	0.0002	0.0002	0.0002	0.0002	0.0002
α_{Ant}	0.0005	0.0115	0.0115	0.0115	0.0115	0.0115
Stability	Bistable	Monostable/persister	Monostable/normal growth	Monostable/normal growth	Bistable	Both

viewed in Table I. According to this paper, the degradation rate of toxin and antitoxin proteins is defined by the following equations:

$$\alpha_{tox} = \frac{ln2}{t_{0.5Tox}} \tag{9}$$

$$\alpha_{Ant} = \frac{ln2}{t_{0.5Ant}} \tag{10}$$

The term $t_{0.5}$ is the normal half-life for toxin and antitoxin. For toxin, $t_{0.5Tox}$ `= 2880 min (2 days)`.[11] For antitoxin, $t_{0.5Ant}$ `= 60 min`. Now the rate of degradation for toxin becomes ~ 0.0002 and the rate of degradation for antitoxin becomes ~ 0.012. These values are consistent with the literature about the type II TAS as antitoxin proteins are more significantly more unstable that toxin proteins. Keeping all other parameters the same, I simulated this new system. Fig. 2 shows that, since antitoxins are so unstable, toxins dominate the system. The system is stable and becomes a persister for seven days. As persisters are such a small fraction of a cell population, it is apparent that other terms play a bigger roll such that antitoxins dominate the cell for most of its lifetime.

I changed the scaling factor for antitoxin, β_{Ant}, in order to balance out the population of antitoxin with the toxin, in my third run. The factor was set to two which represents a production ratio of 1:2 for toxins to antitoxins. Fig. 3 shows that the scaling factor changes the antitoxin population to be twice the toxin population. The system is stable and is a normal growing cell for seven days. This is what most cells look like as persisters only make up a very small percentage of the population. There is still a concentration of toxins that is being consistently neutralized by antitoxins. Unfortunately, increasing the simulation time did not change the stability of the system.

For the next run, I tested various initial concentrations of toxins. I believed that an initial concentration of toxins would change the cell to a persister state. However, the simulation was able to successful neutralize this initial concentration as shown in Fig. 4. In this figure, the intial concentration of toxin protein is 100 nM. Since toxins repress their own production and antitoxins are produced twice as much, this system was able to quickly remove the impact of the toxin proteins.

I began varying the amounts of A_{Half} and T_{Half} to determine the impacts the half-maximal concentration constant has on the system. I ran the simulations while varying both constants between 10 to 100 nM. For most of the runs, the system stayed stable in either the persister or normal growth state. Unfortunately, the values used in the paper by Fasani led to a system that was completely stable. Further testing revealed that there were some values for A_{Half} and T_{Half} that made the system unstable. My first encounter, of the system consistently switching from the growth phase to the persister phase, was when $A_{Half} = 40$ and $T_{Half} = 60$ shown in Fig. 5. The production of toxins stochastically overtakes the production of antitoxins. This cascading effect effectively neutralizes the population of antitoxins and prevents the cell from leaving the persister state. Similar graphs also occurred when: $A_{Half} = 50$ & $T_{Half} = 80$, $A_{Half} = 50$ & $T_{Half} = 90$, and $A_{Half} = 50$ & $T_{Half} = 100$.

While these runs were able to switch consistently from a normal growth state to a slow growing state, there was no run such that the cell did not stay in one state for the whole simulated time. I was looking for a set of values that could cause the simulation to be entirely a normal cell for a run or be an unstable system for another. The values that could cause this phenomenon was discovered when I set $A_{Half} = 60$ and $T_{Half} = 100$. Fig. 6 shows three runs that were relatively stable and three runs that were unstable. From these results, it seems that the half-maximal concentration constant has a strong impact on the stability of the Type II toxin-antitoxin system. A larger half-maximal concentration constant, in this case, indicates that the ribosome has a lower binding affinity to the mRNA. The cell, logically, favors producing antitoxins over toxins to prevent its own death. These results have shown that, even though a cell desires the production of antitoxins over toxins, a concentration of toxins can stochastically repress the production of antitoxins to the point that the cell flips to a persister state. Most cells will never enter this state. However, there is still a small chance that they will and survive while their brethren die.

IV. CONCLUSION

My results elucidates that a bacteria system does not even need to undergo stress to produce persister cells. There is always a small chance that these cells will naturally form to survive a potential disastrous event. The size of the bistable region can be manipulated by varying the half-maximal concentration constants for both the toxin and antitoxin proteins. The production of persisters seems to be stochastic process since persisters make up a small subpopulation of cells. The toxin-antitoxin system might be this process as this system is intrinsically noisy. The concentration of toxins can randomly be neutralized or repress production of antitoxins. This can cause the cell to either continue growing or persist.

While a cell is in the persister state, there is a rare chance that the cell can produces antitoxins to neutralize the abundance of toxins. This causes the cell to recover and become a natural growing cell again which might be the cause of chronic illnesses. However, this behavior could not be explained or observed by the equations I used for the simulations. More research is imperative to understand how the type II TAS can cause the recovery of a persister cell. Hopefully with this research, new antibiotics can be subsequently produced that target persister cells as well as normal growing cells.

ACKNOWLEDGMENTS

I wish to acknowledge David Specht for discussions and suggestions.

[1] J. W. Bigger, TREATMENT OF STAPHYLOCOCCAL INFECTIONS WITH PENICILLIN BY INTERMITTENT STERILISATION, The Lancet **244**, 497 (1944).
[2] H. S. Moyed and K. P. Bertrand, *JOURNAL OF BACTERIOLOGY*, Tech. Rep. 2 (1983).
[3] I. Keren, N. Kaldalu, A. Spoering, Y. Wang, and K. Lewis, Persister cells and tolerance to antimicrobials, FEMS Microbiology Letters **230**, 13 (2004).
[4] K. Lewis, Persister Cells, Annual Review of Microbiology **64**, 357 (2010).
[5] K. Lewis, Persister cells, dormancy and infectious disease, Nature Reviews Microbiology **5**, 48 (2007).
[6] C. Potera, Forging a Link Between Biofilms and Disease, Science **283**, 1837 (1999).
[7] F. Hayes, Toxins-antitoxins: plasmid maintenance, programmed cell death, and cell cycle arrest., Science (New York, N.Y.) **301**, 1496 (2003).
[8] S.-M. Kang, D.-H. Kim, C. Jin, and B.-J. Lee, A Systematic Overview of Type II and III Toxin-Antitoxin Systems with a Focus on Druggability., Toxins **10**, 10.3390/toxins10120515 (2018).
[9] H. S. Deter, R. V. Jensen, W. H. Mather, and N. C. Butzin, Mechanisms for Differential Protein Production in Toxin-Antitoxin Systems., Toxins **9**, 10.3390/toxins9070211 (2017).
[10] D. T. Gillespie, *Exact Stochastic Simulation of Coupled Chemical Reactions*, Tech. Rep.
[11] M. A. Savageau and R. A. Fasani, Unrelated toxin-antitoxin systems cooperate to induce persistence 10.1098/rsif.2015.0130.

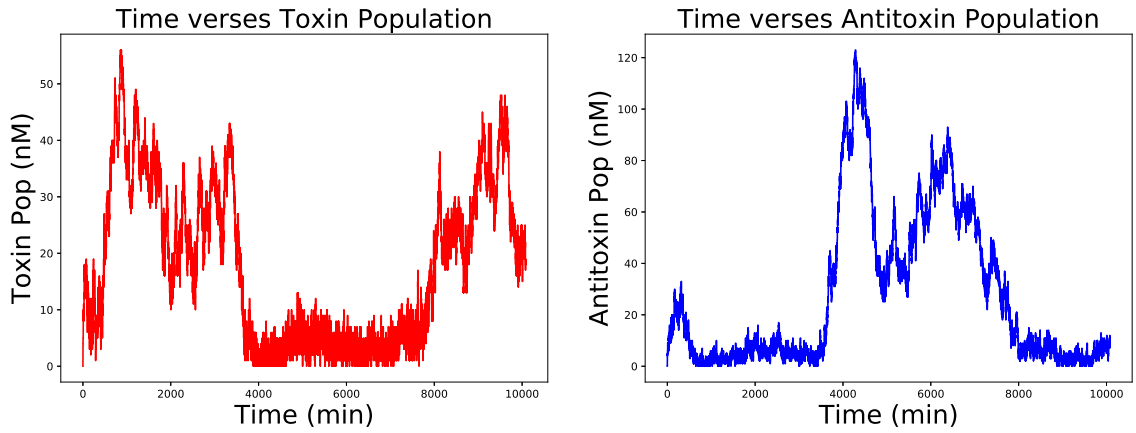

FIG. 1: Run 1 depicts an unstable system with the dominate protein population switching twice. The system goes from persister state to normal growing state to persister state. This plot occurs when the initial values toxin and antitoxin are the same.

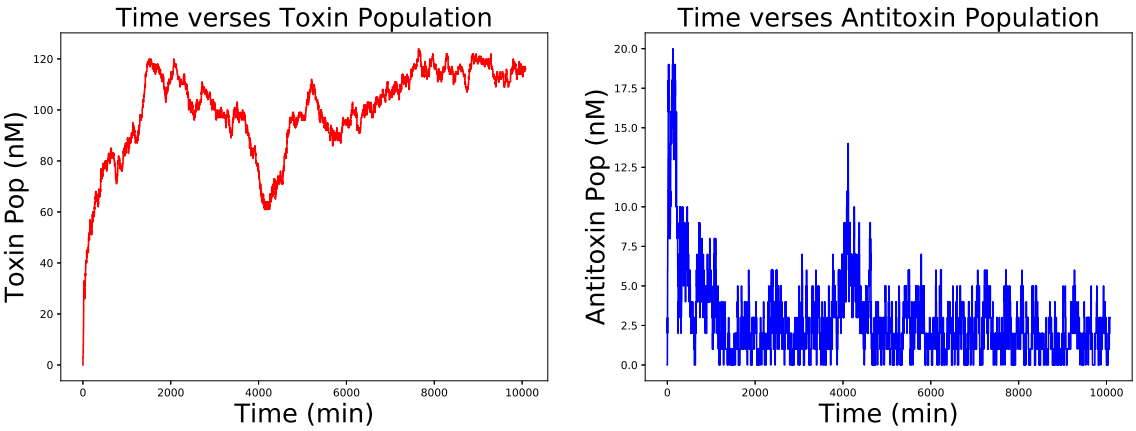

FIG. 2: Run 2 depicts a stable slow growing persister state as toxins completely overtake and repress the production of antitoxins using degradation values from Fasani et al[11].

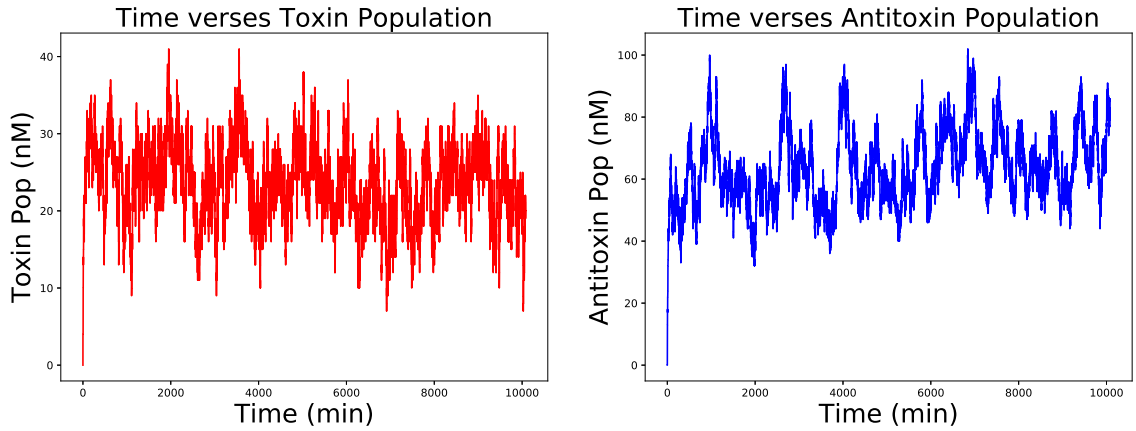

FIG. 3: Run 3 depicts what most normal growing cells experience using a 1:2 production ratio of toxins to antitoxins. There is still a population of toxins, but they are being suppressed by the antitoxin population. Since antitoxins degrade extremely quickly, more antitoxins need to be made compared to toxins.

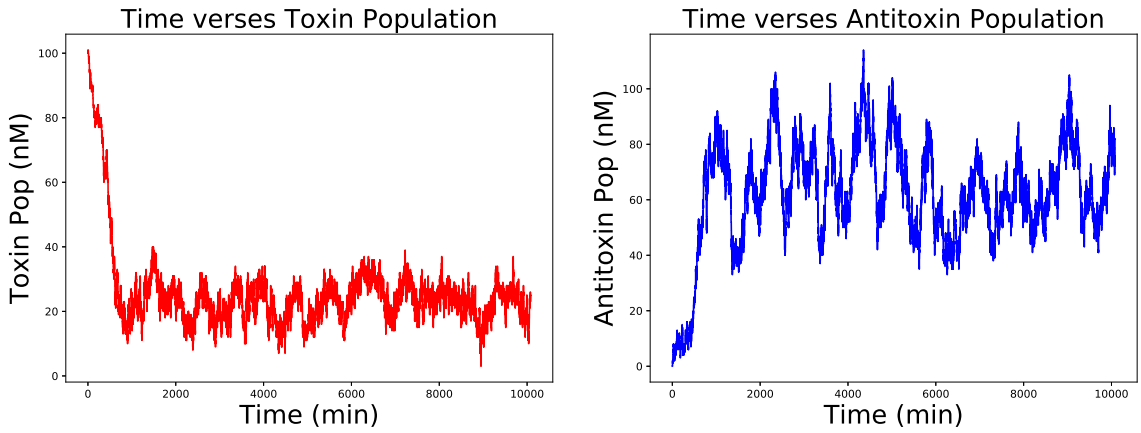

FIG. 4: Run 4 depicts what a cell system, with the same initial values as Run 3, swiftly neutralizing an initial concentration of 100 nM of toxins.

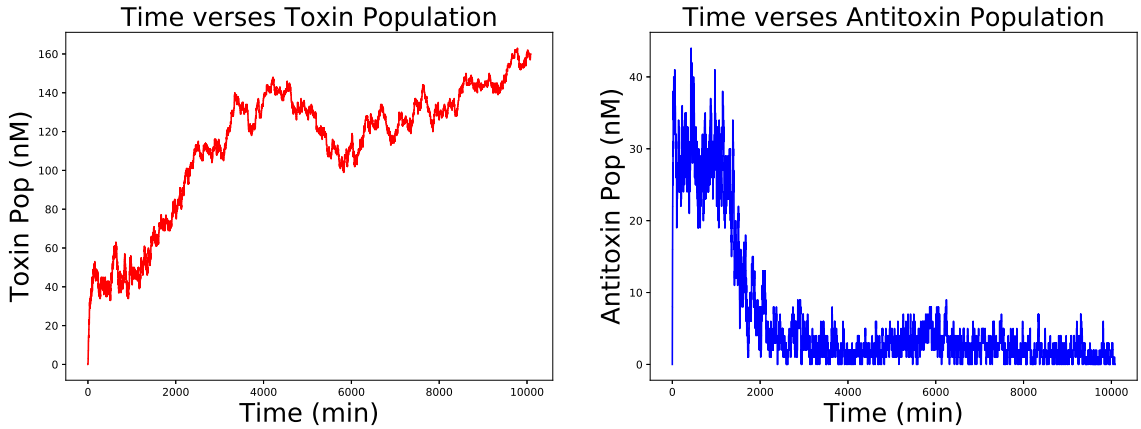

FIG. 5: Run 5 depicts a cell quickly switching from a growth state to the persister state when $\beta_{tox} = 40$ nM and $\beta_{Ant} = 60$ nM.

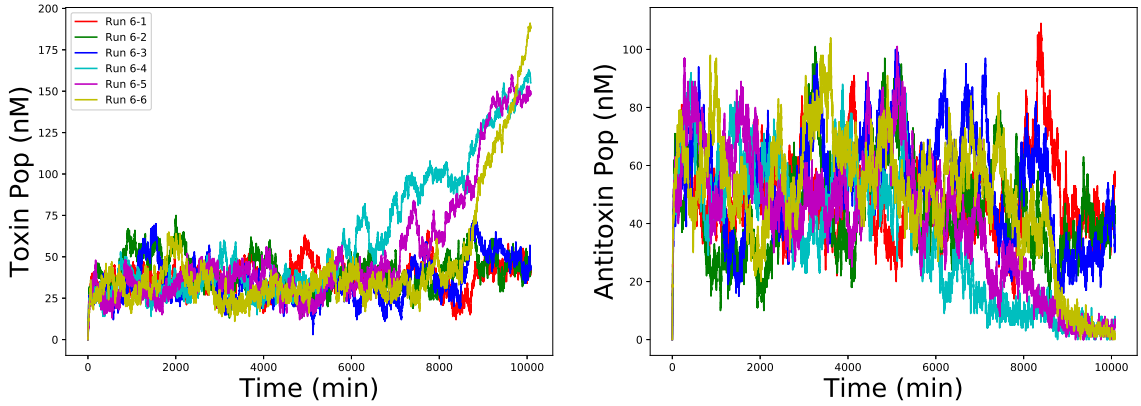

FIG. 6: Run 6 shows six separate simulation with the same initial parameters of $\beta_{tox} = 100$ nM and $\beta_{Ant} = 60$ nM. Three runs (6-1, 6-2, and 6-3) behave akin to a normal growing cell. Antitoxins regulate the toxin concentration accordingly. Three runs (6-4, 6-5, and 6-6) behave as a normal cell, in the beginning, and stochastically transitions into a persister state when toxin concentrations repress the production of new antitoxins.

AAbraham_TA-Model

August 5, 2019

In [1]:
```
#Allan Abraham
#Dr. Lambert
#Gillespie Algorithm Project

#1) Initialization
#2) Monte Carlo
#3) Update
#4) Iterate
```

In [2]:
```
#imports
import numpy as np
import os.path
import ipywidgets as widgets
from IPython.display import display
import matplotlib.pyplot as plt
```

In [3]:
```
def gillespie():
    global t, nAnt, nTox, nNeu, n

    #Calculate propensities

    #Toxin production, neutralized, degradation
    w1 = (BetaTox)/(1+(nTox/Thalf)**n)
    w2 = k*nAnt*nTox
    w3 = alphaT*nTox

    #Antitoxin production, degradation
    w4 = (BetaAnt)/(1+(nTox/Ahalf)**n)
    w5 = alphaA*nAnt

    #total rate
    Rtot = w1+w2+w3+w4+w5

    #generate random numbers
    r1 = np.random.rand()
    r2 = np.random.rand()

    #Time till next reaction
    dt = (1.0/Rtot)*np.log(float(1.0/r1))
    t += dt

    #which reaction occurs
    r2 = r2*Rtot


    #production of toxin
    if r2 < w1:
        nTox += 1

    #production of neutralized pair
    elif r2 < (w1+w2) and nTox != 0 and nAnt != 0:
        nTox -= 1
        nAnt -= 1
```

```
            nNeu += 1

        #degredation of toxin
        elif r2 < (w1+w2+w3) and nTox != 0:
            nTox -= 1

        #production of antitoxin
        elif r2 < (w1+w2+w3+w4):
            nAnt += 1

        #degredation of antitoxin
        elif nAnt != 0:
            nAnt -= 1

        #update populations
        Time.append(t)
        Antpop.append(nAnt)
        Toxpop.append(nTox)
        Neupop.append(nNeu)
```

In [4]:
```
#Four populations: Toxin, Antitoxin, Toxin-Antitoxin Pair (Neutral), Cell
#Change these input parameters

#initial concentrations
#toxin
nTox = 00.0
#antitoxin
nAnt = 00.0
#neutralized pair
nNeu = 0.0

#inital rates

#Rate of neutralized pair formation
k = 0.0005

#initial time
t = 0

#final time
T = 10080*5

#Scaling factor
#Translation factor
BetaTox = 1
#Changes how the protein ratio that is produced
BetaAnt = 2

#Concentration when reaction rate is at half
Thalf = 140

Ahalf = 70

#Rate of toxin degradation
alphaT = np.log(2)/2880 #.0005

#Rate of antitoxin degradation
alphaA = np.log(2)/60 #0.005

#Hill coefficient - number of particles needed for binding
n = 2.0

#Result arrays
Time = []
Toxpop = []
Antpop = []
Neupop = []
```

```
        #Beginning
        Time.append(t)
        Toxpop.append(nTox)
        Antpop.append(nAnt)
        Neupop.append(nNeu)

In [5]: #saving initial concentratoins
        intTox = nTox
        intAnt = nAnt
        intNeu = nNeu

        #Running gillespie Algorithm
        while t < T:
            gillespie()

In [7]: #plotting
        fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(18,6))
        #ax3
        fig.subplots_adjust(wspace=0.2)

        ax1.step(Time, Toxpop, 'r')
        ax1.set_xlabel('Time (min)', fontsize=24)
        ax1.set_ylabel('Toxin Pop (nM)', fontsize=24)
        ax1.set_title('Time verses Toxin Population', fontsize=24)

        ax2.step(Time, Antpop, 'b')
        ax2.set_xlabel('Time (min)', fontsize=24)
        ax2.set_ylabel('Antitoxin Pop (nM)', fontsize=24)
        ax2.set_title('Time verses Antitoxin Population', fontsize=24)
        #Uncomment to add Neutralized Concentration Graph, add ax3 to subplots and change ncols
        = 3
        #ax3.step(Time, Neupop, 'g')
        #ax3.set_xlabel('Time (min)', fontsize=24)
        #ax3.set_ylabel('Neutral Pop (nM)', fontsize=24)
        #ax3.set_title('Time verses Neutral Population', fontsize=24)

        Button = widgets.Button(description = "Save Output?")
        inText = widgets.Text()
        outText = widgets.Text()
        TextFile = None
        epsFile = None
        display(Button)

        #TO SAVE DATA
        #Make sure to change name of the files you want to write to as you could lose previously
        saved data!
        def handle_submit(sender):
            global TextFile, epsFile
            outText.value = inText.value
            TextFile = inText.value +'.txt'
            epsFile = inText.value +'.eps'
            i = len(Time)
            a = 0

            if os.path.isfile(TextFile) or os.path.isfile(epsFile):
                print("File Exists. Please choose different name.")
                return

            F = open(TextFile, 'w+')
            F.write('Maximum Run Time = ' +str(T)+ '\n\n')
            F.write('Initial Concentrations:\nToxin Concentration = ' +str(intTox)+ '\nAntitoxin
        Concentratin = ' +str(intAnt)+ '\nNeutralized pair concentration = ' +str(intNeu)+ '\n')
            F.write('\nRate of neutralized pair formation = ' +str(k)+ '\nRate of toxin
        degradation = ' +str(alphaT)+ '\nRate of Antitoxin Degradation = ' +str(alphaA)+ '\n')
            F.write('\nTranslation Factor of Toxin = ' +str(BetaTox)+ '\nTranslation Factor of
        Antitoxin = ' +str(BetaAnt)+ '\n')
            F.write('\nToxin Half Maximal Concentration = ' +str(Thalf)+ '\nAntitoxin Half
```

```
Maximal Concentration = ' +str(Ahalf)+ '\nHill Coefficient = ' +str(n)+'\n')
    F.write('\n Total of ' +str(i)+ ' events Occurred\n')
    F.write('\nTime\t\t\tNumber of Toxin\t\tNumber of Antitoxins\t\tNumber of
Neutralized Pairs\n')

    while a < i:
        F.write('%-28s %-23s %-10s %24s\n' % (str(Time[a]), str(Toxpop[a]),
str(Antpop[a]), str(Neupop[a])))
        a += 1

    #print("Reaction Rate total\tReaction probability\tNum of Pair\tNum of Tox\tNum of
Ant")
    #print(Rtot, "\t\t", r2, "\t\t", nNeu, "\t\t", nTox, "\t\t", nAnt)


    F.close()
    fig.savefig(epsFile, format='eps', dpi=1000)

    print(TextFile)
    print(epsFile)
    print('Creation Succesful')

def on_button_clicked(b):
    print('Input file name without file extension')

    display(inText)
    inText.on_submit(handle_submit)

Button.on_click(on_button_clicked)
```