2014

Stochastic Differential Equations and Numerical Applications

Matthew Rajotte
Virginia Commonwealth University

Stochastic Differential Equations and Numerical Applications

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science, Virginia Commonwealth University at Virginia Commonwealth University.

by

Matthew Rajotte
Master of Science, Virginia Commonwealth University

Director: Angela Reynolds, Assistant Professor
Department of Mathematics and Applied Mathematics

Virginia Commonwealth University
Richmond, Virginia
May 2014

Acknowledgment

I would like to thank my family and friends for all their support and patience during the course of my thesis. I am grateful to have had the chance to work with my advisor Dr. Angela Reynolds who provided exemplary guidance and help throughout this process, as well as Dr. Cheng Ly, Dr. Rebecca Segal and Dr. Rebecca Heise for graciously serving on my committee.

Contents

List of Figures

Abstract

STOCHASTIC DIFFERENTIAL EQUATIONS AND NUMERICAL APPLICATIONS

By Matthew Rajotte, Master of Science, Virginia Commonwealth University.

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science, Virginia Commonwealth University at Virginia Commonwealth University.

Virginia Commonwealth University, 2014.

Director: Angela Reynolds, Assistant Professor, Department of Mathematics and Applied Mathematics.

We will explore the topic of stochastic differential equations (SDEs) first by developing a foundation in probability theory and Itô calculus. Formulas are then derived to simulate these equations analytically as well as numerically. These formulas are then applied to a basic population model as well as a logistic model and the various methods are compared. Finally, we will study a model for low dose anthrax exposure which currently implements a stochastic probabilistic uptake in a deterministic differential equation, and analyze how replacing the probablistic uptake with an SDE alters the dynamics.

Introduction

Stochastic differential equations (SDEs) are differential equations where stochastic processes represent one or more terms and, as a consequence, the resultant solution will also be stochastic.

For example, a simple model for population growth is given by

$$\frac{dN(t)}{dt} = a(t)N(t)$$

where $N(t)$ is the given population at time t and $a(t)$ is the growth rate.

As is, this is just an ordinary differential equation which can be solved using normal methods. But in nature, $a(t)$ may be subject to some random environmental noise and inputs that are unknown. Thus we can express $a(t)$ as

$$a(t) = r(t) + \sigma(t) \cdot \text{ "noise".}$$

The ordinary differential equation becomes an SDE expressed by

$$\frac{dN(t)}{dt} = N(t)r(t) + N(t)\sigma(t) \cdot \text{ "noise".}$$

The subject of Stochastic Differential Equations lets us make sense of this equation and allows us to derive a solution.

In addition to stochastic analogs of classic differential equations, SDEs are important in filtering problems, stochastic approaches to deterministic boundary value problems, optimal stopping, stochastic control, and mathematical finance [1].

1.1 Mathematical Principals and Brownian Motion

To understand the dynamics of most SDEs and their solutions, it is important to have some knowledge of probability theory as well as some mathematical/statistical principles. Here is a brief introduction to some of these concepts.

Probability Spaces

A σ-*algebra*, $\mathcal{F}$, on a given set Ω is a collection of subsets on Ω where the following properties hold:

(i) $\emptyset \in \mathcal{F}$

(ii) If $A \in \mathcal{F}$, then $A^C \in \mathcal{F}$, where A^C is the complement of A

(iii) If $A_1, A_2, \ldots \in \mathcal{F}$, then $\bigcup_{k=1}^{\infty} A_k \in \mathcal{F}$.

The pair, $(\Omega, \mathcal{F})$, of a set and a σ-algebra is called a measurable space. A function $P : \mathcal{F} \to [0,1]$ is a probability measure on $(\Omega, \mathcal{F})$ such that

(i) $P(\emptyset) = 0$, $P(\Omega) = 1$

(ii) If $A_1, A_2, \ldots \in \mathcal{F}$, then $P(\bigcup_{k=1}^{\infty} A_k) \leq \sum_{k=1}^{\infty} P(A_k)$

(iii) If $A_1, A_2, \ldots \in \mathcal{F}$ and are disjoint ($A_i \cap A_j = \emptyset$ for $i \neq j$), then $P(\bigcup_{k=1}^{\infty} A_k) = \sum_{k=1}^{\infty} P(A_k)$ [1] [3].

The triple of the set, a σ-algebra, and the probability measure, $(\Omega, \mathcal{F}, P)$, is called a probability space. Also, the subsets F of Ω that belong to $\mathcal{F}$ are called *events* and P(F) is interpreted as the probability that the event F will occur.

A random variable X is defined to be a real valued function defined on a sample space [2] and can be classified as either discrete or continuous. A collection of random variables

$X(t)|t \geq 0$ is known as a *stochastic process* and the mapping $t \mapsto X(t,\omega)$ for each $\omega \in \Omega$ is known as a *sample path*. One of the main characteristics of stochastic processes is that if multiple experiments were run, different sample paths would be observed [1][3].

A discrete random variable is one which can take on at most a countable number of possible values. For example, let X be defined as the sum of two fair dice. Then X would be a discrete random variable since it can only take on integer values between 2 and 12.

A probability mass function, p, is defined to be the function that describes the probability that a discrete random variable, X, is exactly equal to some value, say a. This is generally written as

$$p(a) = P\{X = a\}.$$

$p(a)$ is positive for at most a countable number of a values. In other words, if X were to assume one of the values $x_1, x_2, \ldots$, then $p(x_i) > 0$ for $i = 1, 2, \ldots$ and $p(x) = 0$ for any other value of x. Since X must be one of the values of x_i, then

$$\sum_{i=1}^{\infty} p(x_i) = 1.$$

X is said to be continuous if there exists a non negative function $f(x)$, known as the *probability density function* (PDF) of the random variable X, defined for all real x that has the property

$$P\{X \in B\} = \int_B f(x)dx$$

for any set of real numbers, B. In other words, to find the probability that X will be in a

set, one would integrate the probability density function over that set. For example, if one wanted to find the probability that $a \leq X \leq b$ or $X \in [a,b]$ then the equation becomes

$$P\{a \leq X \leq b\} = \int_a^b f(x)\,dx$$

This property also implies that for a continuous random variable, the probability that X equals a specific point a is $P\{\ X = a\} = \int_a^a f(x)dx = 0$. Also, because X must assume a real value, $f(x)$ has the result

$$P\{X \in (-\infty,\infty)\} = \int_{-\infty}^{\infty} f(x)dx = 1.$$

One of the most popular continuous random variable, and one that will be mentioned in the properties of Brownian motion and Wiener processes, is known as a normal random variable, denoted $N(\mu,\sigma^2)$ which means that it has a normal (or Gaussian) distribution with mean μ and variance σ^2. The corresponding PDF is:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}.$$

Expected Value and Variance

The expected value (or expectation) and variance are two important quantities of a random variable.

The expectation, or expected value, of a random variable (denoted $E[x]$) is just as the name suggests. It is the value one would expect in a given experiment. For a discrete

random variable, the expected value is the average of the possible outcomes weighted by the probabilities that X would assume that specific value:

$$E[X] = \sum_{x:p(x)>0} xp(x).$$

Similarly, for a continuous random variable with probability density function $f(x)$ the expected value can be calculated by

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx.$$

Now consider a function of a random variable, say $g(X)$. Since this is simply just another random variable , an expected value can be calculated via:

$$E[g(X)] = \sum_{x:p(x)>0} g(x)p(x)$$

for discrete X, and

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx$$

for continuous X.

Two random variables, X and Y, are said to be independent if $E[XY] = E[X]E[Y]$ provided that both $E[X] < \infty$ and $E[Y] < \infty$.

The variance of a random variable is essentially a value that measures how the probability distribution is spread out. This can be calculated using the expected value by

$$Var(X) = E[(X - E[X])^2]$$

It can be shown that this equation can be put into a useful identity, namely [2]

$$Var(X) = E[X^2] - (E[X])^2.$$

Brownian Motion

In 1828, Scottish botanist Robert Brown observed random and irregular motion of grains of pollen suspended in liquid. This motion was later described the grains randomly colliding with the molecules of the liquid and is referred to as Brownian motion. In 1923, American mathematician Norbert Wiener laid down the mathematical foundation for Brownian motion as a stochastic process, called the Wiener process.

Brownian motion is the basic model for the cumulative effects of pure noise and is denoted as $B(t)$, which indicates the position of a particle at time t. Brownian motion is a stochastic process and has the following properties:

1. For $t > s$, $B(t) - B(s)$ is independent of the past. This is also referred to as independence of increments
2. $B(t) - B(s)$ has a normal distribution with mean 0 and variance $t - s$. In other words, Brownian motion has normal increments
3. $B(t), t \geq 0$ are continuous functions of t.

The initial position in not necessarily defined. A Wiener process is often defined as the standard Brownian motion, which means that those three properties still hold, but now the

initial position is defined to be W(0)=0. From property 2 it can be seen that, if s is taken to be 0 it implies that $W(t)-W(0)$ has $N(0,t)$ distribution (or a normal distribution with mean 0 and variance t).

One more important concept that comes from a Wiener process is the idea of white noise. This is usually denoted $\xi(t)$, and is defined formally as the time derivative of a Wiener process:

$$\xi(t) = \frac{dW(t)}{dt} = W'(t).$$

White noise is typically what one would see as the noise term in an SDE due to the properties of a Wiener process and Itô integrals (which will be discussed later) [4].

1.2 Itô Calculus

With some of the basics of probability theory presented, stochastic calculus can be introduced. Looking back at the noisy equation for population growth introduced earlier

$$\frac{dN(t)}{dt} = N(t)r(t) + N(t)\sigma(t) \text{ "noise"}$$

or in the more general form

$$\frac{dX}{dt} = b(X_t,t) + \sigma(X_t,t)\cdot\text{"noise"}$$

for a random variable X and some functions b, σ. The noise term is represented by a white noise process, $\xi(t)$. Using $\frac{dX}{dt} = \frac{X_{k+1}-X_k}{dt}$ and the definition of the white noise process stated earlier, the equation can be rewritten as

$$X_{k+1} - X_k = b(X_k,t_k)\Delta t_k + \sigma(X_k,t_k)\xi(t)\Delta t = b(X_k,t_k)\Delta t_k + \sigma(X_k,t_k)\Delta W_t$$

where W_t is Brownian motion, or a Wiener process. Extrapolating to X_k from X_0 gives:

$$X_k = X_0 + \sum_{j=0}^{k-1} b(X_j, t_j)\Delta t + \sum_{j=0}^{k-1} \sigma(X_j, t_j)\Delta W_j.$$

Now, using integral notation, this becomes

$$X_t = X_0 + \int_0^t b(X_s, s)\, ds + \int_0^t \sigma(X_s, s)\, dW_s.$$

For a more rigorous proof that $\int_0^t f(w,s)dW_s(w)$ does, in fact, exist in a certain sense, see Chapter 3 in Oksendal [1].

The first integral term is deterministic and can be solved using normal calculus rules. The question now becomes how to make mathematical sense of the integral with respect to Brownian motion, $\int_0^t \sigma(X_s, s)dW_s$. Naturally, one would assume that it is the limit defined by

$$\lim_{|\pi|\to 0} \sum_{k=0}^{n-1} \sigma_{t_k}(W_{\pi_{k+1}} - W_{\pi_k})$$

where $0 = \pi_0 < \pi_1 < \ldots < \pi_n = t$ is a partition of $[0,t]$ and $|\pi| := max_k(\pi_{k-1} - \pi_k)$ represents the refinement of this partition. For deterministic functions, this represents the Riemann Integral and the choice for t_k does not matter as $|\pi| \to 0$. Though due to the properties of Brownian motion, this is not the case for stochastic functions. For example, consider $\int W_t dW_t$ and take $t_k = \pi_{k+1}$, the limit becomes

$$\lim_{|\pi_{k+1}-\pi_k|\to 0} \sum_{k=0}^{n-1} W_{\pi_{k+1}}(W_{\pi_{k+1}} - W_{\pi_k}) = \lim_{|\pi_{k+1}-\pi_k|\to 0} \sum_{k=0}^{n-1} W_{\pi_{k+1}}^2 - W_{\pi_{k+1}} W_{\pi_k}.$$

Letting $t_k = \pi_k$, the limit is

$$\lim_{|\pi_{k+1}-\pi_k|\to 0}\sum_{k=0}^{n-1} W_{\pi_{k+1}}W_{\pi_k} - W_{\pi_k}^2.$$

Subtracting this form the first one with $t_k = \pi_{k+1}$

$$\lim_{|\pi_{k+1}-\pi_k|\to 0}\sum_{k=0}^{n-1} W_{\pi_{k+1}}^2 - 2W_{\pi_{k+1}}W_{\pi_k} - W_{\pi_k}^2 = \lim_{|\pi_{k+1}-\pi_k|\to 0}\sum_{k=0}^{n-1}(W_{\pi_{k+1}} - W_{\pi_k})(W_{\pi_{k+1}} - W_{\pi_k})$$

and taking the expectation

$$E\big[\lim_{|\pi_{k+1}-\pi_k|\to 0}\sum_{k=0}^{n-1}(W_{\pi_{k+1}} - W_{\pi_k})(W_{\pi_{k+1}} - W_{\pi_k})\big] = \sum_{k=0}^{n-1}(\pi_{k+1} - \pi_k) = t.$$

Therefore, regardless of how refined the partition becomes, the difference between the two will have a constant expectation. This means the choice of t_k matters and the classic Riemann integral cannot be used. This also implies that W_t is nowhere differentiable.

To avoid this problem, for X_t continuous, $\mathrm{E}\left[\int_0^T X_t^2\,dt\right] < \infty$, and X_t non-anticipating (meaning for every realization and n, X_n is known at time n), the Itô integral defines $\int X_t dW_t$ as the limit of

$$S_\pi = \sum_{k=0}^{n-1} X_{t_k}(W_{\pi_{k+1}} - W_{\pi_k})$$

but not a limit in a normal sense, but rather saying the random variable $S = \int X_t dW_t$ satisfies

$$E[|S - S_\pi|^2] \xrightarrow{|\pi|\to 0} 0.$$

In other words, the Itô integral is defined by:

$$\int X_t dW_t := \sum_{k=0}^{n-1} X_{t_k}(W_{t_{k+1}} - W_t)$$

where t_k is taken to be the left endpoint.

It is worth noting another popular stochastic integral called the Stratonovich integral. This is where X_{t_k} is replaced with the midpoint, $X_{\frac{t_k+t_{k+1}}{2}}$. The main reason the Stratonovich integral may be used is because it demonstrates that it obeys the traditional rules of integration, under quite relaxed conditions. Even with this, the Itô integral is typically favored more since it is non-anticipatory (the Stratonovich integral is not) and it preserves the Martingale property (which is for a sequence of random variables $X_1, \ldots, X_n$, $E(X_{n+1}) = X_n$).

Also, another notable outcome is that both X_t and $\int X_t dW_t$ can be seen as Hilbert spaces and the integration is just a mapping from one to another. Therefore, the following is known as the Itô isometry [5]

$$E\left[\int_0^t X_s^2\, ds\right] = E\left[\left(\int_0^t X_s dW_s\right)^2\right].$$

Itô Formula

Although there is a definition for the Itô integral, it is a hassle to go through the whole limit process. Therefore, it is much more convenient to use what is known as the Itô formula.

Let X_t be an Itô process defined by

$$dX_t = u dt + v dW_t$$

and let $g(x,t)$ be twice differentiable. Then $Y_t = g(X_t,t)$ is also an Itô process:

$$dY_t = \frac{\partial}{\partial t}g(X_t,t)dt + \frac{\partial}{\partial x}g(X_t,t)dX_t + \frac{1}{2}\frac{\partial^2}{\partial x^2}g(X_t,t)\cdot(dX_t)^2$$

where $(dX_t)^2 = (dX_t)\cdot(dX_t)$ is computed by the rules $dt\cdot dt = dt\cdot dW_t = dW_t\cdot dt = 0$ and $dW_t\cdot dW_t = dt$.

To see an example of this, consider the integral $\int_0^t W_s dW_s$. Take $X_t = W_t$ and $g(x,t) = \frac{1}{2}x_t^2$. Then $Y_t = \frac{1}{2}W_t^2$ and by the Itô formula

$$dY_t = \frac{\partial g}{\partial t}dt + \frac{\partial g}{\partial x}dW_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}(dW_t)^2 = 0 + W_t dW_t + \frac{1}{2}dW_t^2 = W_t dW_t + \frac{1}{2}dt.$$

Therefore

$$d(W_t^2) = W_t dW_t + \frac{1}{2}dt$$

and integrating both sides from $[0,t]$ with some arithmetic gives

$$\int_0^t W_s dW_s = \frac{1}{2}W_t^2 - \frac{1}{2}t.$$

There is also a way to define "integration by parts" for a stochastic integral using the Itô formula which is defined as

$$\int_0^t f(s)dW_s = f(t)W_t - \int_0^t W_s df_s$$

where in this case it is important that f is bounded as well as continuous for this to hold. [1]

Application

Recall the model for population growth

$$\frac{dN_t}{dt} = a_t N_t$$

where $a_t = r_t + \alpha W_t$, W_t is white noise, and α, $r_t = r$ are constants. The model now becomes equivalent to

$$dN_t = rN_t dt + \alpha N_t dW_t \tag{1.1}$$

by the Itô interpretation where, in this case, $\sigma(t,x) = \alpha x$. This can further be rewritten as

$$\frac{dN_t}{N_t} = rdt + \alpha dW_t.$$

Therefore

$$\int_0^t \frac{dN_s}{N_s} = rt + \alpha W_t. \tag{1.2}$$

To calculate $\int_0^t \frac{dN_s}{N_s}$, the Itô formula is implemented with $g(t,x) = \ln(x)$. This yields

$$d(\ln(N_t)) = \frac{1}{N_t} dN_t - \frac{1}{2N_t^2}(dN_t)^2. \tag{1.3}$$

Using (1.1) for dN_t and the properties that say $dt \cdot dt = dt \cdot dW_t = 0$ and $dW_t \cdot dW_t = dt$, $(dN_t)^2$ can be written as

$$(dN_t)^2 = r^2 N_t^2 dt^2 + 2r\alpha N_t^2 dt dW_t + \alpha^2 N_t^2 dW_t^2 = \alpha^2 N_t^2 dt.$$

Now (1.3) becomes

$$d(\ln(N_t)) = \frac{dN_t}{N_t} - \frac{1}{2N_t^2}\alpha^2 N_t^2 dt = \frac{dN_t}{N_t} - \frac{\alpha^2}{2}dt.$$

or

$$\frac{dN_t}{N_t} = d(\ln(N_t)) + \frac{\alpha^2}{2}dt$$

Now this equation can be integrated from 0 to t and can be set equal to (1.2). Doing this gives us

$$\ln\left(\frac{N_t}{N_0}\right) + \frac{\alpha^2}{2}t = rt + \alpha W_t$$

Solving for N_t, the final solution to the stochastic population model is defined as

$$N_t = N_0 e^{(r-\frac{\alpha^2}{2}t+\alpha W_t)}. \tag{1.4}$$

This section was adapted from Oksendal [1].

Numerical Methods of SDEs

Euler-Maruyama

Generally speaking, explicit solutions for initial value problems of ordinary differential equations are impossible to find. One of the most popular ways to approximate the solutions to these problems is to use the Euler method. This method takes a differential equation in the form, $\dot{x} = f(t,x)$ with initial condition $x(0) = x_0$ and approximates it using the formula

$$y_{i+1} = y_i + f(t_i, y_i)\Delta t$$

for a time discretization $t_0 < t_1 < t_2 < \ldots < t_n$. Where $y_0 = x_0$ and $\Delta t_i = t_{i+1} - t_i$ or simply the time step being used for the approximation.

Due to the stochastic nature of SDEs, the Euler method cannot be used directly, though there is a stochastic analog of Euler's method which is known as the Euler-Maruyama method [6]. The steps to derive this method are similar to the way one would derive the non stochastic Euler method.

For an SDE in the form $dX_t = a(t,X_t)dt + b(t,X_t)\xi(t)$, where $\xi(t)$ represents white noise, the goal is to find a formula that will approximate the solution at the next time step, X_{n+1}, given that the previous time step, X_n, is known.

First, dX_t can be defined as

$$dX_t = \frac{\Delta X_t}{\Delta t}$$

which implies

$$\Delta X_t = dX_t \Delta t.$$

The original equation, $dX_t = a(t,X_t)dt + b(t,X_t)\xi(t)$, can be substituted in for dX_t. This then yields

$$\Delta X_t = (a(t,X_t)dt + b(t,X_t)\xi(t))dt.$$

Using $\Delta X_t = X_{n+1} - X_n$ and the fact white noise is defined as $\frac{\Delta W}{dt}$ where $\Delta W = W_{n+1} - W_n$ is a Wiener increment . This gives the final formula of

$$X_{n+1} = X_n + a(t_n, X_n)\Delta t + b(t, X_n)\Delta W_t$$

for a time discretization $t_0 < t_1 < t_2 < \ldots < t_n$ and initial condition X_0. Here since Brownian motion is normally distributed with variance t, ΔW_t will have standard deviation of $\sqrt{dt}$ and can be represented as just $r_n(\sqrt{dt})$, where r_n is a random number from a normal distribution [6].

Fokker-Planck Equation

Another useful equation for numerical applications of SDEs is the Fokker-Planck (or Kolmogorov forward) equation. This takes an SDE in the form

$$dX_t = h(X_t,t)dt + \sqrt{2g(X_t,t)}dW_t$$

where $h(X_t,t)$ is the drift term and $g(X_t,t)$ is the diffusion coefficient, and gives a formula for the derivative with respect to time of the PDF, $f(x,t)$, of the random variable X_t:

$$\frac{\partial}{\partial t} f(x,t) = \frac{\partial^2}{\partial x^2}[g(x,t)f(x,t)] - \frac{\partial}{\partial x}[h(x,t)f(x,t)].$$

This equation is for the one-dimensional random variable but can be extended to a multi-dimensional case defined by

$$d\mathbf{X_t} = \mathbf{h}(\mathbf{X_t}, t)dt + \mathbf{b}(\mathbf{X_t}, t)d\mathbf{W_t}$$

in which case $\mathbf{X_t}$ is an n-dimensional random variable and $\mathbf{W_t}$ is an m-dimensional Wiener process.

The Fokker-Planck equation is now given by

$$\frac{\partial}{\partial t} f(\mathbf{x}, t) = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}[\mathbf{h_i}(\mathbf{x}) f(\mathbf{x}, t)] + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial}{\partial x_i \partial x_j}[D_{ij}(\mathbf{x}) f(\mathbf{x}, t)]$$

where $D_{ij}(x,t) = \frac{1}{2} \sum_{k=1}^{m} b_{ik}(\mathbf{x}, t) b_{jk}(\mathbf{x}, t)$ is the diffusion tensor.

The Fokker-Planck equation is commonly used for variables describing a macroscopic but small subsystem. For example, position and velocity for the Brownian motion of a particle, a current in an electrical circuit, and the electrical field of a laser [7].

2.1 Application

Population Growth Model

Recalling, again, the stochastic model for population growth, $dN_t = rN_t dt + \alpha N_t dW_t$, simulations can be run using the Euler-Maruyama method and then compared to the exact solution (1.4) which was obtained using the Itô formula.

For Figures 2.1 and 2.2, MATLAB was used to run simulations of the exact solution and the Euler-Maruyama approximation on the equation

$$dN_t = 0.3N_t dt + 0.2N_t dW_t$$

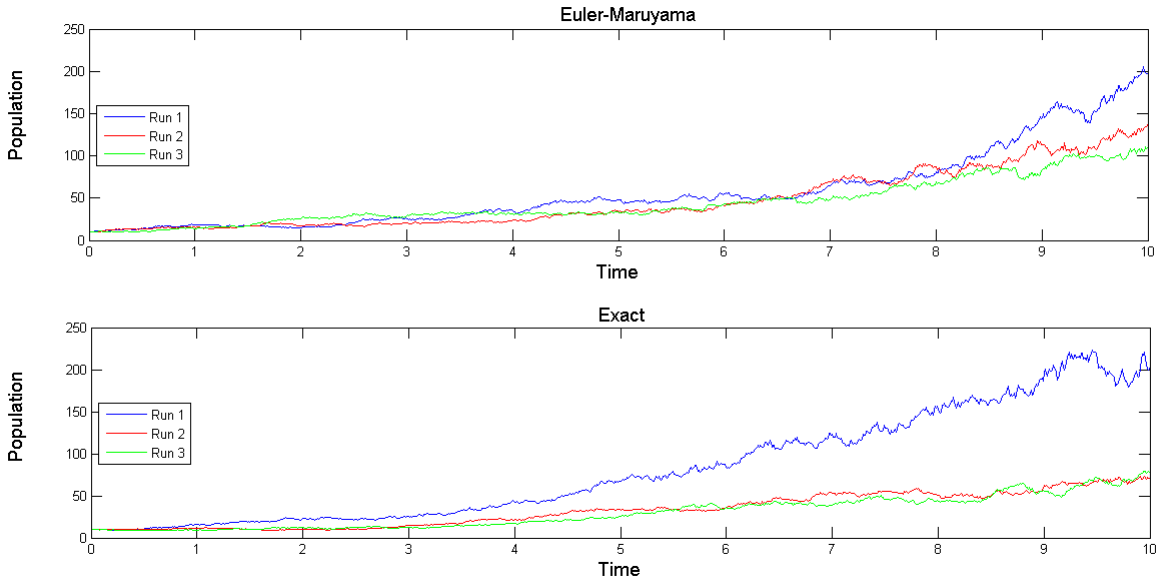

Figure 2.1: Three independent runs of the stochastic model for population growth.

over the time interval [0,10] with dt=0.01 and $X_0 = 10$. The parameter values, $r = 0.3$ and $\alpha = 0.2$, were arbitrarily chosen to run simulations.

In Figure 2.1, three separate runs of the exact solution, $N_t = N_0 e^{\left((0.3-\frac{0.2^2}{2})t+0.2W_t\right)}$, are graphed together. It can be seen that due to the stochastic nature of the model that separate runs with the same parameter values can produce very different results. Due to this, it is generally beneficial to run simulations numerous times and average it to get an overall view of how the system behaves. This is shown in Figure 2.2 where averages were taken after each 10, 100, and 1000 runs of both the exact solution and the Euler-Maruyama approximation and with more runs, the averages of the two converge.

Logistic Growth Model

Another differential equation that looks at how a population changes over time but takes into account the carrying capacity is the logistic growth model. The simplest form of this

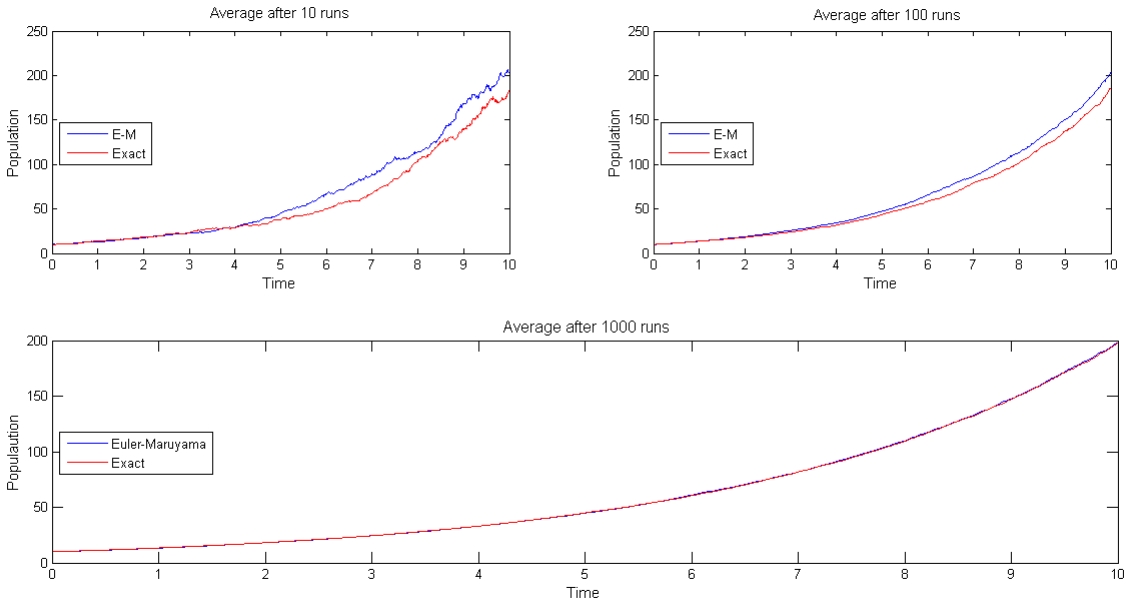

Figure 2.2: Runs for basic population model averaged and compared

equation is:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{k}\right) + \sigma\xi(x)$$

where r is the growth rate, k is the carrying capacity, and the $\sigma\xi(x)$ term is the additive white noise term that makes this equation an SDE.

Using the arbitrary values $r = 1$ and $k = 4$, this equation without noise would rise to a steady state value of $N = 4$. But as shown in Figure 2.3, when the noise is taken into account, the solution for N will rise to $N = 4$ and then "bounce" around that solution, and the level of the noise will determine how severity of this bounce.

A closer look can be taken at the equations after a long time period, say $t = 15000$. At this point, the equation would be in the noisy steady state for a generous amount of

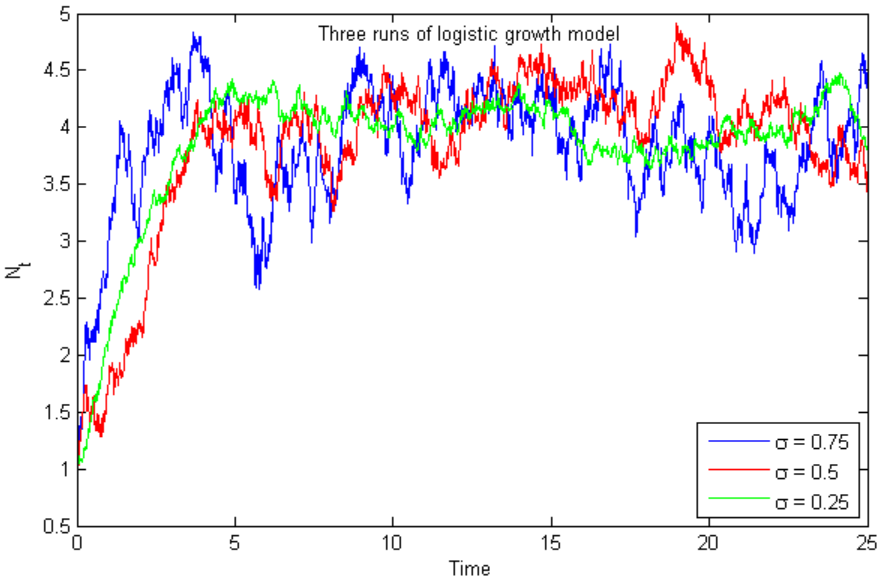

Figure 2.3: Three runs of the logistic growth model using the Euler-Maruyama approximation

time. Figure 2.4 looks at one of these long time runs, after which the different values that N achieves are binned and plots a histogram that essentially is the steady state of the probability density function (PDF) of N

The Fokker-Planck equation can now be used to get an equation of how this PDF changes in time. Doing this yields

$$\frac{\partial f}{\partial t} = \frac{\sigma^2}{2}\frac{\partial^2}{\partial x^2}f(x) - \frac{\partial}{\partial x}\left((x - \frac{x^2}{4})f(x)\right).$$

To be able to compare this to the histograms in Figure 2.4 which are the PDF at a steady state, $\frac{\partial f}{\partial t}$ is set to equal 0. Doing this and integrating both sides with respect to x, the equation becomes

$$\frac{2}{\sigma^2}K = \frac{d}{dx}f(x) - \frac{2}{\sigma^2}(x - \frac{x^2}{4})f(x)$$

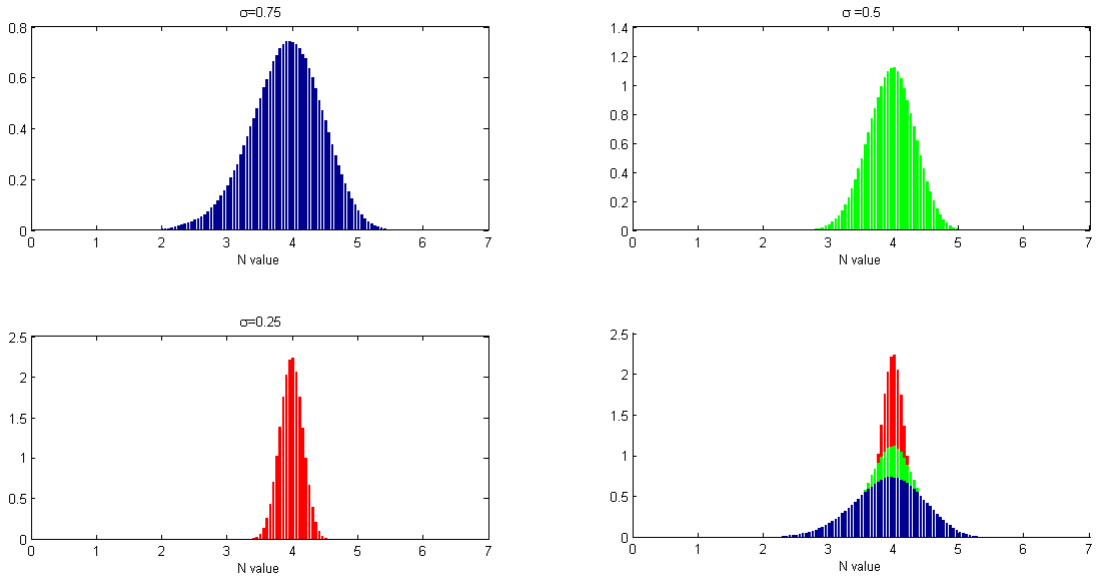

Figure 2.4: Histograms of Eulers method
These histograms are effectively PDFs of the steady state, so integrating over an interval will give the probability of being in that interval. Top left: Histogram of 'steady state' with $\sigma = 0.75$. Top right: Histogram of 'steady state' with $\sigma = 0.5$. Bottom left: Histogram of 'steady state' with $\sigma = 0.25$. Bottom right: All combined

for some constant K. An integrating factor of

$$e^{(-\frac{2}{\sigma^2}\int_0^x rs-\frac{s^2}{4})} = e^{-\frac{2}{\sigma^2}\left(\frac{x^2}{2}-\frac{x^3}{12}\right)}$$

is used to get a solution to this new first order, ordinary differential equation. Multiplying and integrating both sides will now give

$$\frac{2}{\sigma^2}K\int e^{-\frac{2}{\sigma^2}\left(\frac{x^2}{2}-\frac{x^3}{12}\right)}+C = f(x)e^{-\frac{2}{\sigma^2}\left(\frac{x^2}{2}-\frac{x^3}{12}\right)}$$

where C is a new integrating constant. The solution is now solved with K set equal to 0. This eliminates $\int e^{-\frac{2}{\sigma^2}\left(\frac{x^2}{2}-\frac{x^3}{12}\right)}$ which does not have an analytic solution. The final equation left for the PDF of this SDE at the steady state is:

$$f(x) = Ce^{\frac{2}{\sigma^2}\left(\frac{x^2}{2} - \frac{x^3}{12}\right)}$$

where C is the constant that would make the property $\int f(x) = 1$ hold.

Considering the case when $\sigma = 0.5$, the equation becomes $f(x) = Ce^{4x^2 - \frac{2}{3}x^3}$. To find the correct value for the constant, one would numerically integrate this function and determine the C value that ensure $\int f(x) = 1$ to holds. But because of the x^2 and x^3 in the exponential term, numerical integration techniques seemed to be unstable. In order to "stabilize" this some manipulation was done to the constant. C was split into the product of two constants, say C_1 and C_2, and C_2 was subtracted in the exponent. Thus the equation turned into

$$f(x) = C_1 e^{\frac{2}{\sigma^2}\left(\frac{x^2}{2} - \frac{x^3}{12}\right) - \ln(\frac{1}{C_2})}.$$

This allowed for $\ln(\frac{1}{C_2})$ to be chosen as some value to help produce a stabilized approximation to the integral. Since there is already a graph to compare to, the value for C_2 can be chosen as anything that would help and any difference in C_2 would only affect C_1, but would not end up affecting the product of the two. After this, it is determined that the original constant had the value $C \approx 6.0888 \times 10^{-9}$. Figure 2.5 plots this equation for $f(x)$ along with the histogram for $\sigma = 0.5$ that was plotted in Figure 2.4. As, expected, the two match very well.

Analytic solutions to the Fokker-Planck equation, like the Itô SDE, are impossible to calculate, in general. An equivalent mathematical formulation of the steady-state Fokker-Planck equation is a boundary value problem (BVP). Once again using $\sigma = 0.5$, the Fokker-Planck equation after setting $\frac{\partial f}{\partial t} = 0$ becomes

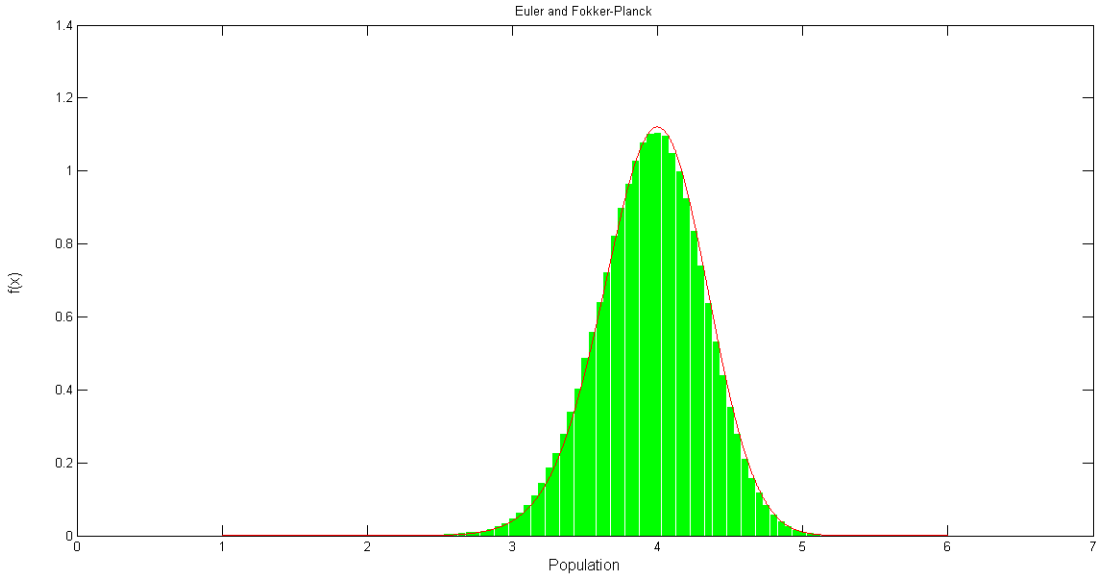

Figure 2.5: Euler Histograms vs Analytic Fokker-Planck

Again, both of these functions represent the PDF of the steady state. Comparison of the histogram taken from the long run of the Euler-Maruuyama method and the steady-state PDE given by the Fokker-Planck equation.

$$0 = \frac{1}{8}\frac{d^2}{dx^2}f(x) - \frac{d}{dx}\left(\left(x - \frac{x^2}{4}\right)f(x)\right)$$

or equivalently,

$$0 = \frac{1}{8}\frac{d^2}{dx^2}f(x) - \left[\left(x - \frac{x^2}{4}\right)f'(x) + \left(1 - \frac{1}{2}x\right)f(x)\right]$$

when a simple chain rule is used for the second term on the right hand side. Now this second order equation can be converted to a system of first order equations by setting some variable $y = f'(x)$, therefore $y' = f''(x)$. The equivalent system is now in the form

$$\begin{bmatrix} f'(x) \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 8-4x & 8x-2x^2 \end{bmatrix} \begin{bmatrix} f(x) \\ y \end{bmatrix}$$

. with boundary conditions that represent a no flux scenario of $0 = 8y - x(1 - \frac{x}{4})f(x)$ at the boundaries. From the histograms in Figure 2.4 the boundaries are taken to be $x = 1$ and $x = 6$.

The solution of this BVP was calculated in XPP and exported to MATLAB so the integral could be scaled to 1. Figure 2.6 shows the plot of the scaled solution and compares again to the histogram of the steady state of the long run Euler-Maruyama method. As expected, figure shows that both methods of finding this steady state PDF produce the same results.

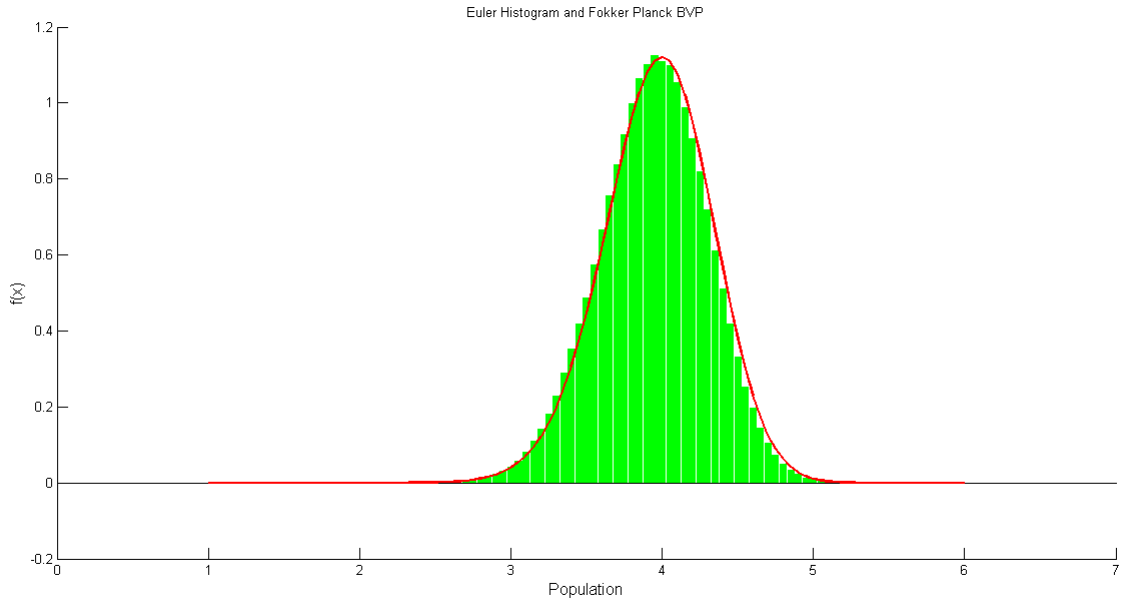

Figure 2.6: Euler Histogram and Fokker-Planck BVP
Comparison of the solution to the Fokker-Planck BVP and the histogram from Figure 2.4

Low Dose Anthrax Model

In this chapter, a model for low dose anthrax absorption, which is a component of Dr. Reynold's research on low dose exposure to anthrax, is analyzed to see if numerical applications of SDEs can be applied. Contribution to this project is being made by considering the effects that are made by replacing the current rules which implements a stochastic probabilistic uptake into a deterministic differential equation with one that uses a noisy term in the differential equation, thus making it an SDE.

Preliminaries and Original Model

When a person breaths in anthrax spores, the spores are dispersed throughout the lung where antigen presenting cells (APCs) uptake the spores and transport them to the lymph node. The main role of these cells are to present insults to the immune system and trigger a larger immune response. While inside the APCs spores can germinate, releasing bacteria in the cell, which replicate and can cause the APCs to lysis (break open) and release bacteria into the blood.

A set of differential equation is used to model the dynamics during the transportation from the lung to the lymph node for the APCs. We assume that both macrophages and dendritic cells can uptake the spores. Therefore we model the dynamics for spores that are internalized by macrophages and dendritic cells as separate variables (S_m and S_d , respectively). We assume that spores are more likely to enter a macrophage and that dendritic

cells hold fewer spores than the macrophages. Additionally, dendritic cells do not phagocyte (kill/destroy) as well as macrophages. If the spores germinate and cause the cell to lysis, we increase the extra cellular bacteria, B_e.

The differential equations that model the number of spores in each the dendritic cells and macrophages, as well as the amount of extracellular bacteria, are:

$$S_d' = -\mu_d S_d - k_1 S_d$$
$$S_m' = -\mu_m S_m - k_1 S_m$$
$$B_e' = k_1 n_b (S_m + S_d) + g B_e (1 - B_e / B_{e_{max}})$$

where μ_d and μ_m are the destruction of the spores by the dendritic and macrophage cells, respectively, k_1 is the rate at which cells lysis, n_b is the scaling factor for bacteria to spore ratio, g is the growth rate of the bacteria, and $B_{e_{max}}$ is the carrying capacity for bacteria.

Each time the probabilistic rules are called, they determine the uptake of each external spores and whether internalized spores leave the lung changing the levels of S_m and S_d variables. The fate of the spores in the lung are determined by these rules every 3000 time steps (where $dt = \frac{1}{360000}$).

Figure 3.1 shows the resultant plot of the number of total spores in the system where the total time is taken to be 200 hours and the fate of the spores were added to S_m and S_d every 3000 time steps.

These rules are such that the following hold:

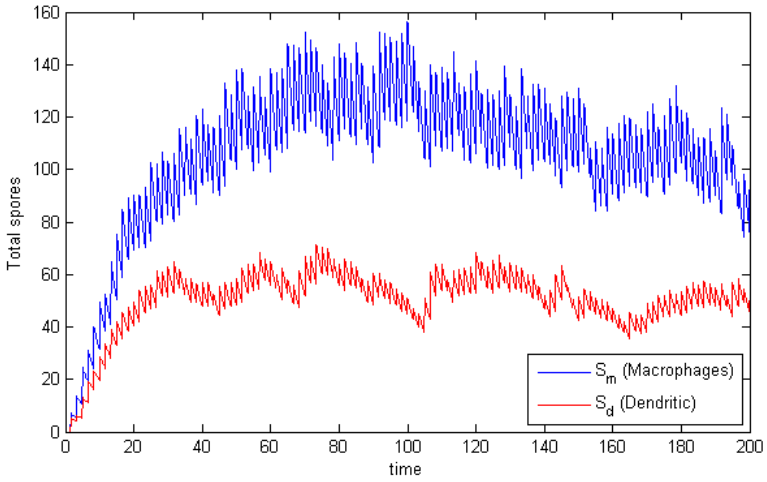

Figure 3.1: Single run of transport model

The plot for the total spores in the system for both macrophages and dendritic cells for a one run simulation.

- The macrophages are closer to the cells, therefore they are more likely to find a given spore than the dendritic cells. Thus for each external spore, it is determine if it will be picked up, and if so, which individual cell gets that spore.

- A max load is determined for each APC at the beginning of the simulation. Dendritic cells have a max load that is lower than the macrophages max load. For these simulations, it is assumed that there are the same number of each cell type, but this can be changed.

- A cell is more likely to leave the lung when it is full, but they can leave before they are full.

- All the spores leaving the lung at a given time window are summed for each APC group and become the increases in S_m and S_d.

Adaptations to the Model

Figure 3.2 shows the same equation with the same parameters averaged over 500 runs. It

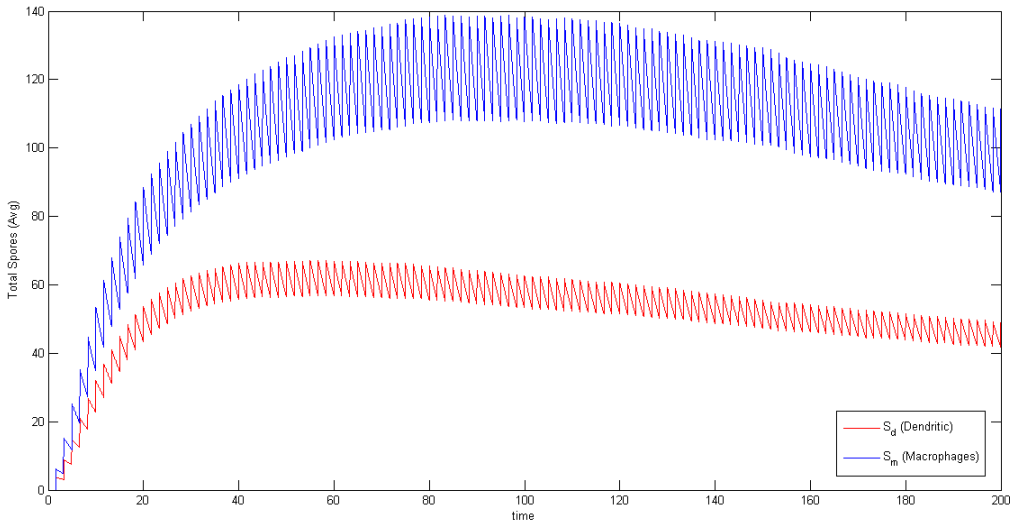

Figure 3.2: Average Total Spores
This figure shows the average of the total spores model after 500 runs.

can be seen that this graph somewhat exhibits some of the properties one would see from the average graph of an SDE.

To try and explain this behavior, a closer look is taken at the dynamics of stochastic uptake of the external spores due to the equations that determine a spores fate in the cells. Figure 3.3 shows, in essence, a scatter plot of the spores arriving into the system. At first look, there seems no pattern to these incoming spores, but looking at the average of these arriving spores over 1000 runs (Figure 3.4) it becomes quite clear there is some pattern emerging.

The next step is to somehow express this average as a continuous function. Since the fate of the spores are determined every 3000 time steps, this means that the points in Figure 3.4 do not represent the amount of spores arriving at that instantaneous time, but rather the *total* amount arriving over the 3000 time steps. To account for this, one can just divide all the points by 3000 and this will yield an approximation for the amount of spores arriving

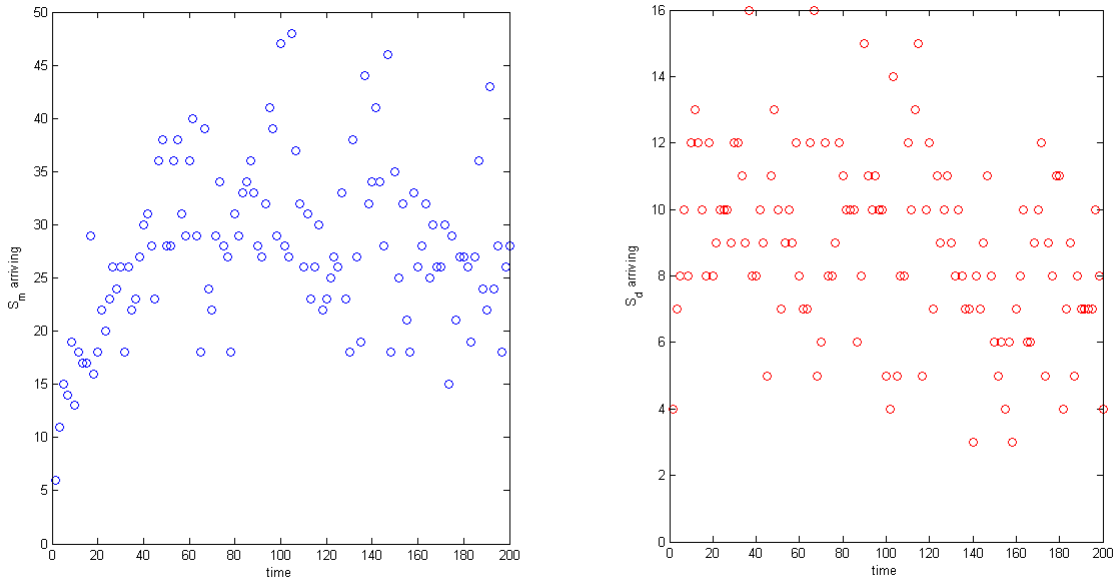

Figure 3.3: Plot of spores arriving
These plots show the stochastic input of spores arriving into the differential. Left: spores arriving from macrophage. Right: spores arriving from dendritic cells

at that exact time only. After that is done, a polynomial can be fitted to the data and a continuous function that represents the amount of spores coming into the system can be obtained. Figure 3.5 shows just that, a 4^{th} degree polynomial representing the amount of spores at any given time fitted to this data.

Creation of SDE

With an equation for the mean of the spores, and the average standard deviation, an SDE model can be implemented. The differential equations for S_m and S_d with the addition of a noise term results in:

$$S_d' = -\mu_d S_d - k_1 S_d + \sigma_1(S_d,t)\xi(t)$$

$$S_m' = -\mu_m S_m - k_1 S_m + \sigma_2(S_m,t)\xi(t)$$

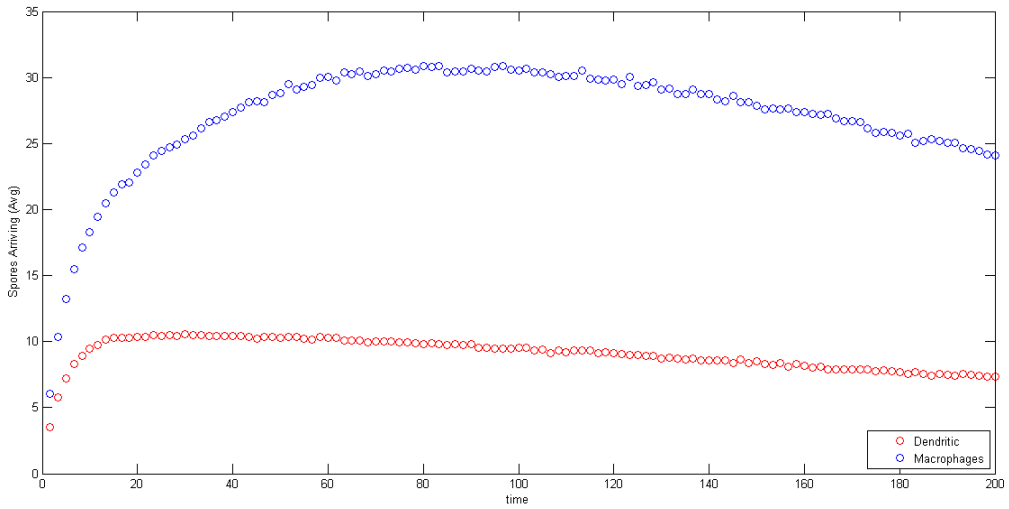

Figure 3.4: Average Arrival
The average at a given time step of 1000 runs are plotted and a pattern emerges.

where σ_1 and σ_2 are functions that have the mean of there respective 4^{th} order polynomial and average standard deviation found in Figure 3.4, and $\xi(t)$ is white noise which is used so that the Euler-Maruyama method can be used to do simulations.

Recalling the formula for the Euler-Maruyama method

$$\mathbf{X_{n+1}} = \mathbf{X_n} + a(t_n, (\mathbf{X_n})\Delta t + b(t, X_n)\Delta \mathbf{W_t},$$

in this case, the function a is just the original differential equation with no noise. Unfortunately, the equation for the noise, or b in the formula, is not explicitly known. But mean and standard deviation of this function are known and can be taken into account essentially in the ΔW_t "term."

Since ΔW_t is a Wiener increment (Brownian motion) it has a normal distribution of mean 0 and a standard deviation of $\sqrt{dt}$. Thus normally, when implementing the Euler-Maruyama

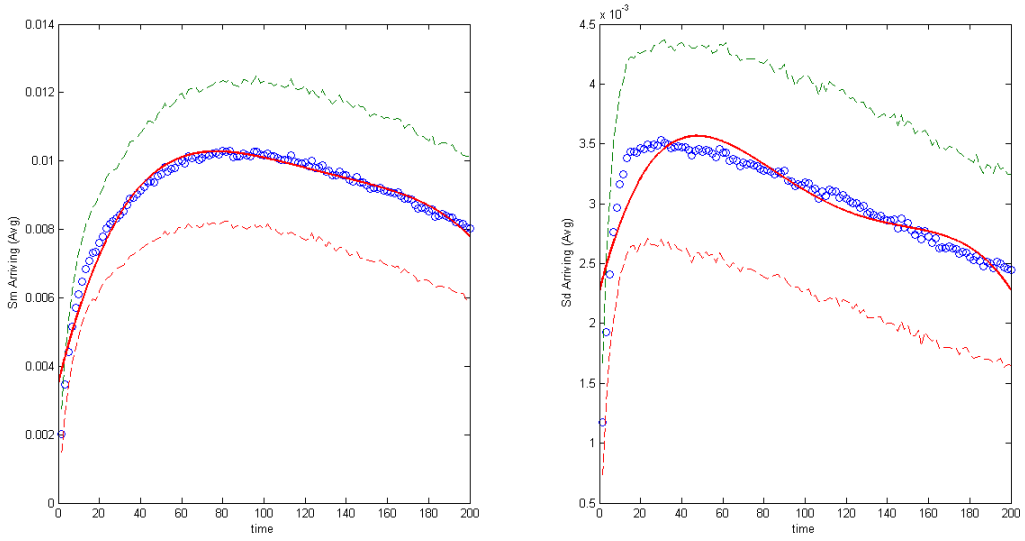

Figure 3.5: Polynomial Fit for Averaged Arriving Spores
After the points from Figure 3.4 were divided by 3000, they were plotted along with their standard deviation. Then a polynomial of degree 4 was fit to the data representing the mean.

method in MATLAB for example, one would code this as (randn$\cdot\sqrt{dt}$). To adjust the mean of this, one would just add the desired mean to this (randn$\cdot\sqrt{dt}$) and adjust the standard deviation by multiplying (randn$\cdot\sqrt{dt}$) by th new standard deviation. So now focusing on the equation of S'_m, the way one would "code" this using what is known of the mean and standard deviation would be

$$S_{m_{n+1}} = S_{m_n} + (-\mu_m S_{m_n} - k_1 S_{m_n})dt + (P_4(t) + \gamma \cdot \text{randn} \cdot \sqrt{dt})$$

where $P_4(t)$ is the 4th order polynomial that represents the mean and γ represents the average standard deviation.

Figure 3.6 shows these simulations for both S_m and S_d on $t \in [0,200]$ averaged over 500 runs and compared to the average obtained from the original equation in Figure 3.2. It can be seen that this SDE adaptation for this problem produces a good fit to the data generated by the original equation that just had a stochastic probabilistic uptake into a

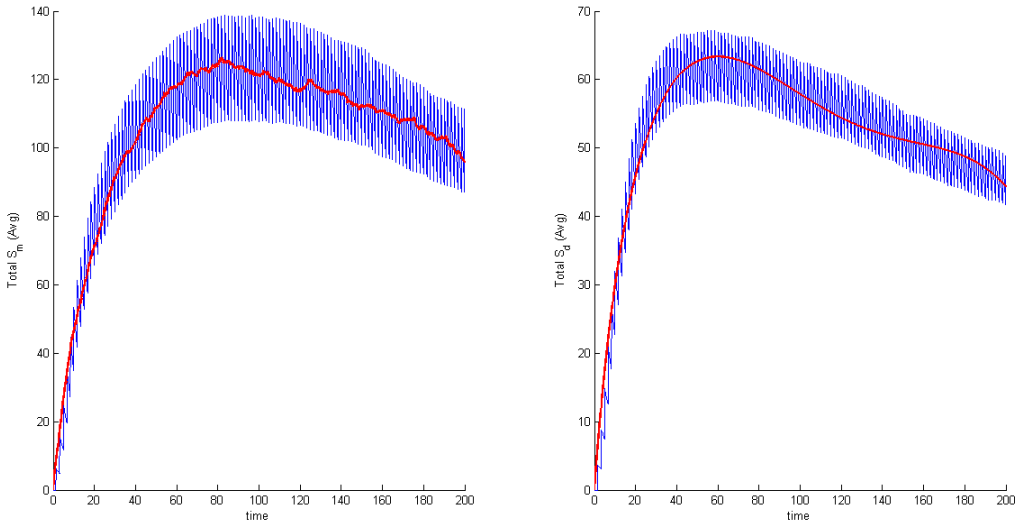

Figure 3.6: Averaged Euler-Maruyama Plot
Averages form Figure 3.2 compared with the averages of the Euler-Maruyama simulations.

deterministic differential equation. The SDE method would prove to be useful when the model is being used for lower doses. In this scenario, the variance of the deterministic method (the spikes and valleys in Figure 3.2) would become problematic. That is the case because these transport equations will eventually feed into equations that model the response of the immune system and the variance being a high percentage of the overall number of spores will cause problems getting a consistent response over time.

It is important to note that these simulations and comparisons were done with a certain set of parameters. So if any were to changes to these initial conditions, the whole process would have to be repeated to obtain the polynomial describing the average as well as the standard deviation. Future work on these equations could include finding out how all the initial conditions affect the mean representing polynomial or even finding a formula for quicker derivation of it.

Bibliography

[1] Bernt Oksendal, *Stochastic Differential Equations*, 6th ed, Springer-Verlag, 2013

[2] Sheldon Ross, *Introduction to Probability Models*, 10th ed, Elsevier, 2010

[3] Lawrence Evans *An Introduction to Stochastic Differential Equations*, 2011. Web. Retrieved from http://www.math.uh.edu/ torok/math_6397_SDE/Evans_SDE.course_v1.2_2011.08.pdf

[4] Fima Klebaner, *Introduction to Stochastic Calculus with applications*, 3rd ed, Imperial College Press, 2012

[5] Xin Tong, *A Crash Course for Stochastic Calculus*, NYU, 2013. Web, Retrieved from http://www.cims.nyu.edu/ chennan/CrashCourseonStochasticCalculus.pdf

[6] Peter Kloeden, *Numerical Solutions to Stochastic Differential Equations*, Springer-Verlag, 1992

[7] H. Risken, *The Fokker-Planck Equation: methods of solution and applications*, Springer-Verlag, 1989

Appendix A

Euler-Maruyama Method

```
%This MATLAB function impletents the Euler-Maruyama method on an SDE %


function [Xt] = eulersde( x0, t0, tf, dt, f, g)

% x0 -- initial value of random variable
% t0 -- start time
% tf -- end time
% dt -- time increment
% f -- function associated with the deterministic part of the SDE
% g -- funciton associated with the stochastic part of the SDE

t=t0:dt:tf;  %sets time interval

n=max(size(t)); %determines number of iterations

sqrtdt=sqrt(dt);
```

```
Xt=zeros(n,1);  %initializes vector and sets the initial condition

Xt(1)=x0;

for i=1:n-1
    Xt(i+1)=Xt(i)+f(Xt(i))*dt+g(Xt(i))*randn*sqrtdt;
end

end
```

Appendix B

Exact Solution for Population Growth Model

```
%This MATLAB function computes sample run of the exact solution of the
%population growth model that was found using the Ito formula

function [Nt] = exactsde(x0, t0, tf, dt, r, alpha)


t=t0:dt:tf; %sets time interval

n=max(size(t)); %determines number of iterations

sqdt=sqrt(dt);

Bt=zeros(n,1); %Bt is a vector representing Brownian motion (random walk)
Nt=zeros(n,1); %Nt is the random variable vector

Nt(1)=x0; %Sets the initial value (Note: initial value for Bt is 0)

for i=1:n-1
    Bt(i+1)=Bt(i)+sqdt*randn;
```

```
    Nt(i+1)=Nt(1)*exp((r-(1/2*(alpha)^2))*t(i)+alpha*Bt(i+1));
end

end
```

Appendix C

Averages of Population Growth Model

```
%This MATLAB script creates Figure 2.2 which averages the exact solution
%and the Euler-Maruyama approximation of the population growth model.

sum=0;              %Initializes sums that will be used to average
sum1=0;



dt=.01;             % Sets time step and iterations
n=10/dt;

for i=1:n
    s=eulersde(10,0,10,dt,@(x) 0.3*x,@(x) .2*x);
    sum=sum+s;                    %Runs and stores in a single vector of
                                  %both the exact and E-M approximation
    s1=exactsde(10,0,10,dt,0.3,.2);
    sum1=sum1+s1;

    if i==10      %Produces and plots average after 10 runs
```

```
        z=sum/10;
        z1=sum1/10;
        subplot(2,2,1)
        plot(t,z,t,z1,'r')
    end


    if i==100                %Average and plots after 100 runs
        z=sum/100;
        z1=sum1/100;

        subplot(2,2,2)
        plot(t,z,t,z1,'r')
    end

end

t=0:dt:10;

y=sum/n;
y1=sum1/n;                %Plots final average (1000 in this case)

subplot(2,2,3)
plot(t,y,t,y1,'r')
```

Appendix D

Histograms for Euler-Maruyama Method of Logistic Equation

```
%This MATLAB script runs the long term Euler-Maruyama approximation for
%the logistic growth model and creates a histogram

dt=.01;

r=1;
k=4;                %Parameters for the specific equation

x0=4;
t0=0;
tf=15000;           %Set initial conditions and final time

hold on

for i=1:3
    sigma=.75-.25*(i-1);  %Changes the noise level for different runs

    f=@(x) r*x*(1-(x/k));
    g=@(x) sigma;    %Sets deterministic and stochastic equations
```

```
s1=eulersde(x0,t0,tf,dt,f,g); %Runs approximation

x1=linspace(1,6,101);
N=histc(s1,x1);             %Sets and bins N values

n=sum(N);

if i==1
    subplot(2,2,1)
    bar(x1,N/n*20) %Creates histogram and nomalizes integral to 1

    subplot(2,2,4)
    bar(x1,N/n*20)

elseif i==2
    subplot(2,2,2)
    bar(x1,N/n*20,'g')

    s2=s1;

    subplot(2,2,4)
    bar(x1,N/n*20,'y')

elseif i==3
```

```
        subplot(2,2,3)
        bar(x1,N/n*20,'r')

        subplot(2,2,4)
        bar(x1,N/n*20,'r')
    end

end
```

Appendix E

Fokker-Planck BVP for the Logistic Growth Model

```
# This is the input for XPP for the Fokker-Planck BVP of the logistic
#equation with addictive noise

# define the 2 parameters, a,b
par a=1,b=6

# now do the right-hand sides
F'=Y
Y'=(8-4*t)*F+(8*t-2*t^2)*Y

# and finally, boundary conditions
# First we want V(0)-V0=0
bdry 8*Y-t*(1-t/4)*F

# We also want aV(1)+bVX(1)=0
bdry 8*Y'-t*(1-t/4)*F'

# Note that the primes tell XPP to evaluate at the right endpoint
@ total=5, bell=0
```

Appendix F

Low Dose Anthrax Codes–Original Transport Code

```
%This MATLAB function is the actual model being used by Dr. Reynolds in
%her low dose anthrax research that simulates the number of spores being
%transported

function [data,NewIC, total_Sd, total_Sm, Si, Se, d, m ,full_D, full_M]
      =transport(Se, ic, t0,tfinal, total_d, total_m,
      maxload_d, maxload_m)
%tic
% set numerical timestep

h=(tfinal-t0)/360000;

%initialize variables and plot vectors

x(1,:)=ic;
xoutput(1,:)=ic;
t(1)=t0;
toutput(1)=t0;
```

```
s=2; %counter for plotting

% determine the total iterations of forward euler needed.
n=(tfinal-t0)/h;

%intialize the total number of spore within macrophages and dendritic
%cells released from the lung each call to sporefate
%(Sm and Sd, respectively) and total realesed for each cell
%population to zero.

Sm=0;
Sd=0;

total_Sd=0;
total_Sm=0;

% initilizing full vectors. These store the total spore instead "full"
% dendrite and macrophage cells
full_D=[];
full_M=[];

% call function to create dmload vector- this contains the max
%spore uptake
% for each cell. For each dendritic cells (there are total_d
%dendritic cells
% available to consume spores) a random integer is chosen between
```

```
%1 and
% maxload_d and stored in dmload. For each macrophage (there are
%total_m
% macrophages available to consume spores) a random integer is
%chosen between 1 and
% maxload_m and stored in dmload.

dmload=create_dmload(total_d, total_m, maxload_d, maxload_m);

% setup the initial vector of internal spores- all macrophages and
% dendritic cells are initialized to contain zero.

Si=zeros(1,length(dmload));

% initial d (the counter for remaining dendritic cells) to the total
% dendricit cells initial in the system (total_d).
d=total_d;

for i=2:n
    if mod(i,3000)==0

    [Se, Si, dmload, d, Sd, Sm, full_D, full_M]
         =sporefate(Se,dmload,Si,full_D, full_M, d);

    x(i-1,1)=x(i-1,1)+Sd;
    total_Sd=total_Sd+Sd;
```

```
    x(i-1,2)=x(i-1,2)+Sm;
    total_Sm=total_Sm+Sm;
    end

% input of underfilled macrophages and dendrite cells
    % add probabilty of release a function of load

%determine right hand sides of the differential equations
r=dxtransport(t(i-1),x(i-1,:));

%forward Euler method
i

x(i,:)=x(i-1,:)+h*r';
t(i)=t(i-1)+h;

%update vectors for plotting

if mod(i,10)==0
    xoutput(s,:)=x(i,:);
    toutput(s)=t(i);
    s=s+1;
end

end
```

```
toutput=toutput';

%plot output of the numerical approx.

plot(toutput,xoutput(:,1),'-');ylabel('Sd Total');xlabel('time');figure;
plot(toutput,xoutput(:,2),'-');ylabel('Sm Total');xlabel('time');figure;
plot(toutput,xoutput(:,3),'.');ylabel('Extracellular Bacteria');
           xlabel('time');

%plot(toutput,xoutput(:,6),'*');ylabel('TA');xlabel('time');

% setup up output and sets NewIC to be the final output so they can
%easily be used to continue the sim.
data=[toutput,xoutput];
NewIC=xoutput(end,:);

m=length(dmload)-d;
```

Appendix G

Low Dose Anthrax Codes – Sporefate

```
function [Se, Si, dmload, d, Sd,Sm, full_D, full_M]
          =sporefate(Se,dmload,Si,full_D, full_M, d)

%Given the extracelluar spore count (Se), load of each macrophage and
%dendritic cells (dmload, a vector of available spore spots with in the
%D and M population), Si (a vectore which contains the number of already
%internalized spores in each D and M cell), and the number of Dendritic
%cells, d, this function determines the number of Spores reamoved from
%the lung via
%dendritic cells (Sd) and macrophages (Sm) at each call. It always
%return the new adjusted Se, Si, dmload and d.

%Si=zeros(1,length(dmload))
%d=5;

pi=.999999;  %1-pi is the probablity that an individual spore meets
             with a Dendritic cell or a Macrophage

p_leave=.01; % probabilty an occupied leaves before it is full
```

```
p_leave_full=.5; % probabilty full cell leaves

Sd=0; % Initialize Sd for each step
Sm=0; % Initialize Sm for each step

Se_removed=0; % Initialize Se_removed for each step, this will determine
              %the total spores lost local during each run

 % We check whether each spore is taken up.

pref_m=4;%strength of the preference for macrophages, must be an integer

for i=1:Se % We check whether each spore is taken up.

    Pi=1-pi^(length(dmload));%the more total cells (length(dmload))
                             the more likely the spore is consumed.

    r1=rand(1);

    if r1<Pi % if the spore is taken up we determine whether it enters a
             D or M cell and if this causes that cell to meet in
             max capacity.

        r2=randi([1,d+pref_m*(length(dmload)-d)],1); % determines which
       %cell it enters and uploads dmload and Si for the associated
       %component.
```

```
if r2>d
    r2=mod(r2, length(dmload)-d)+d+1;
end

dmload(r2)=dmload(r2)-1;
%dmload

Si(r2)=Si(r2)+1;
Se_removed=Se_removed+1;

r3=rand(1); %random number used to determine if full cells leave

if dmload(r2)==0

% if the cell reaches max the spores and may move from the Si
%component to either Sd (if component r2 is less than d)

    %or Sm (otherwise) and that cell is removed from the cells
    %capable of update.

    if r2<=d

        full_D=[full_D Si(r2)];
        Si(r2)=0;
        d=d-1;
```

```
            elseif r2>d
                full_M=[full_M Si(r2)];
                Si(r2)=0;
            else
                error('error in extending your full_cell vectors')
            end

            % adjust dmload and Si when a Cell becomes full

            for k=r2:length(dmload)-1
                dmload(k)=dmload(k+1);
                Si(k)=Si(k+1);
            end
            dmload=dmload(1:end-1);
            Si=Si(1:end-1);
        end

    end

end

Se=Se-Se_removed; % update the Se population

%Check to see if unfilled cells are leaving
r4=rand(1,length(dmload));
```

```
Sd_not_max=0;
Sm_not_max=0;

i=1;

while i<=length(r4)

    if (r4(i)<p_leave && Si(i)~=0)

        if i<=d
            Sd=Sd+Si(i);
            Sd_not_max=Sd_not_max+Si(i);
            d=d-1; % remove one D cell from the local population
            else

            Sm=Sm+Si(i);
            Sm_not_max=Sm_not_max+Si(i);
        end

            % adjust dmload and Si when a Cell leaves the local
            population

        for k=i:length(dmload)-1
            dmload(k)=dmload(k+1);
            Si(k)=Si(k+1);
```

```
            r4(k)=r4(k+1);
        end

        dmload=dmload(1:end-1);

        Si=Si(1:end-1);
        r4=r4(1:end-1);
    end

     i=i+1;
end

%Check to see if filled cells are leaving, info stored in
full_cells_M and full_cells_D

r5=rand(1,length(full_D));
r6=rand(1,length(full_M));

j=1;

while j<=length(r5)

    if r5(j)<p_leave_full

       Sd=Sd+full_D(j);
```

```
                % adjust dmload and Si when a Cell leaves the
                local population

        for k=j:length(full_D)-1
            full_D(k)=full_D(k+1);
            r5(k)=r5(k+1);
        end

            full_D=full_D(1:end-1);
            r5=r5(1:end-1);
    end

     j=j+1;
end

ji=1;

while ji<=length(r6)

    if  r6(ji)<p_leave_full
       Sm=Sm+full_M(ji);

                % adjust dmload and Si when a Cell leaves the
                local population

        for k=ji:length(full_M)-1
```

```
            full_M(k)=full_M(k+1);
            r6(k)=r6(k+1);
        end
            full_M=full_M(1:end-1);
            r6=r6(1:end-1);
    end
     ji=ji+1;
end
end
```

Appendix H

Low Dose Anthrax Codes – Modified Transport

```
%This MATLAB function is a slghtly modified version of Dr. Reynold's
%transport equation that only focuses on only the macrophages and
%dendretic
%cells and allows for simulations to be run multiple times and averages
%and standard deviations to be obtained

%For more detials of the parts that are similar, see the original.



for k=1:1000
    k
    Se=10000;       %reintialize total number of spores

% set numerical timestep
h=(tfinal-t0)/360000;

%initialize variables and plot vectors
x(1,:)=ic;
```

```
xoutput(1,:)=ic;
t(1)=t0;
toutput(1)=t0;
s=2; %counter for plotting

% determine the total iterations of forward euler needed.
n=(tfinal-t0)/h;



Sm=0;
Sd=0;
total_Sd=0;
total_Sm=0;

% initilizing full vectors. These store the total spore instead "full"
% dendrite and macrophage cells
full_D=[];
full_M=[];



dmload=create_dmload(total_d, total_m, maxload_d, maxload_m);

% setup the initial vector of internal spores- all macrophages and
% dendritic cells are initialized to contain zero.

Si=zeros(1,length(dmload));
```

```
% initial d (the counter for remaining dendritic cells) to the total
% dendricit cells initial in the system (total_d).
d=total_d;
j=1;

for i=2:n
    if mod(i,3000)==0

    [Se, Si, dmload, d, Sd, Sm, full_D, full_M]
            =sporefate(Se,dmload,Si,full_D, full_M, d);

    x(i-1,1)=x(i-1,1)+Sd;

    total_Sd=total_Sd+Sd;

    x(i-1,2)=x(i-1,2)+Sm;
    total_Sm=total_Sm+Sm;

    SdArrive(j,k)=Sd;       %adds current run of arriving spores to a
    SmArrive(j,k)=Sm;       %matrix containing all of the previous runs

    if k==1
    tArrive(j)=t0+h*i;          %determines time spores arrive
    end
```

```
    j=j+1;

    end

%determine right hand sides of the differential equations

r=dxtransport1(t(i-1),x(i-1,:));

%forward Euler method
x(i,:)=x(i-1,:)+h*r';
t(i)=t(i-1)+h;

%update vectors for plotting
if mod(i,10)==0

    xoutput(s,:)=x(i,:);
    toutput(s)=t(i);
    s=s+1;
end
end
end

SmAvg=mean(SmArrive');
SdAvg=mean(SdArrive');

SmActual=SmAvg/3000;
```

```
SdActual=SdAvg/3000;   %Calculates average and standard dev and
                       %estimates actual continuous arrival of spores
SmDev=std(SmArrive');
SdDev=std(SdArrive');

SmDevP=SmActual+(SmDev/3000);
SmDevN=SmActual-(SmDev/3000);

SdDevP=SdActual+(SdDev/3000);
SdDevN=SdActual-(SdDev/3000);

toutput=toutput';

c1=polyfit(tArrive,SmActual,4);       %finds a polynomial fit for the
c2=polyfit(tArrive,SdActual,4);       %arriving spores

f_Sm=@(x) c1(1)*x.^4+c1(2)*x.^3+c1(3)*x.^2+c1(4)*x+c1(5);
f_Sd=@(x) c2(1)*x.^4+c2(2)*x.^3+c2(3)*x.^2+c2(4)*x+c2(5);

tfit=0:.01:200;

Smfit=f_Sm(tfit);
Sdfit=f_Sd(tfit);

%plot output of the numerical approx.
plot(tArrive,SmAvg,'o')
```

```
plot(tArrive,SdAvg,'o')
plot(tArrive,SmActual,'o',tArrive,SmDevP, '--',tArrive,SmDevN,'--',
        tfit,Smfit,'r')
plot(tArrive,SdActual,'o',tArrive,SdDevP, '--',tArrive,SdDevN,'--',
        tfit,Sdfit,'r')
```

Vita

Matthew David Rajotte was born July 24, 1989 in Warwick, Rhode Island. He graduated from Ocean Lakes High School in Virginia Beach, Virginia in 2007. He received his Bachelor of Science in Mathematics from Virginia Polytechnic Institute and State University in 2011.