Comparison of two numerical methods for stochastic delay differential equations and the relationship between bifurcation approximations and step length

Nia Edmunds

September 26, 2014

Abstract

We give introductions to delay differential equations, stochastic differential equations, numerical approximations, Brownian motion and Ito calculus, stability and bifurcation points and Lyapunov exponents. Using these methods we replicate the calculations in the paper by Neville J. Ford & Stewart J. Norton, entitled "Noise-induced changes to the behaviour of semi-implicit Euler methods for stochastic delay differential equations undergoing bifurcation". We present our results that correspond to some of the tables and equations presented in their paper. We then apply the same methodology using a Milstein numerical method with the same parameters and random distributions and compare these results with our findings from the Euler-Maruyama scheme.

We find that the Milstein scheme exhibits the same relational behaviours between the bifurcation approximations from the Lyapunov exponents and step length as was presented in Ford and Norton's paper for the Euler-Maruyama scheme, we also find that the Milstein scheme maintains its greater accuracy up to and including the bifurcation approximation.

Contents

1 Introduction

We will investigate whether the findings of Ford and Norton in their paper entitled "Noise-induced changes to the behaviour of semi-implicit Euler methods for stochastic delay differential equations undergoing bifurcation" [1] hold for the higher order Milstein scheme, in particular the relationship between step length and the approximation of the bifurcation point in stochastic delay differential equations (SDDE) and the resulting relational equations. We suspect there is an equally strong relationship when using the Milstein scheme but will this be the same or similar to the Euler-Maruyama or significantly different?

Ford and Norton's paper is based on Norton's PHD thesis [2] where much of the references and methodology for this paper come from. Initially we intend to follow Norton's methodology and confirm his findings for the Euler-Maruyama scheme and then apply the same methodology and conditions to using the Milstein scheme which will give us a direct comparison from our own results. This means that any deviation in our results should be purely down to the change in the numerical method used.

We begin by discussing the theory behind the methods used, starting with an introduction to delay differential equations (DDE) and how the stability regions of these equations can be calculated numerically from their characteristic equations. We discuss the numerical methods for approximating DDE and derive an Euler scheme using matrices to store and calculate the history caused by the delay in the equations. We then introduce Brownian motion and Ito calculus which allows us to incorporate stochastic terms into a differential equation and apply numerical methods. We then demonstrate the Euler-Maruyama method for stochastic differential equations (SDE). Following a brief discussion on the accuracy and order of numerical methods we then introduce the Milstein scheme. We discuss bifurcation points and using Lyapunov exponents to estimate them and some of the limitations of this method that justify why we have chosen to neglect the investigation of varying the noise levels that Norton conducted as part of his work.

The remainder of this paper is then dedicated to presenting our methodology, results and observations. Using Norton's parameters we run simulations for the Euler-Maruyama scheme, we show that we had comparable results before then applying the same techniques and parameters to the Milstein scheme.

2 Delay differential equations

It is assumed that the reader is familiar with ordinary differential equation (ODE) and the methods of solving them, in particularly for linear ordinary differential equations. ODE's are not always an accurate representation for some models. A method for introducing some more realistic behaviour into a mathematical model is by using delay variables, forming delay differential equations (DDE). These account for situations where there is a time delay between a change in a variable and the effect on a system. Normal ODE's assume everything depends on the current state, DDE's allow for systems where a change at another state (usually time) can influence the current or next state. A typical example of this is the 'adjusting the water temperature in the shower' example, where there is a slight delay after adjusting the taps that the effects are then observed. The effects of incorporating a delay into mathematical model are numerous, some of them include; an initial function needs to be specified instead of an initial value, they generate infinitely dimensional systems, when computing them the history has to be stored, discontinuities can propagate and the systems can be prone to breaking down unexpectedly and exhibiting chaotic behaviours that weren't present in an ODE version of a similar type.

Delay differential equations can be used to model many different things which involve some kind of lag, time delay or feedback loop. They appear in many areas of modelling in medicine and biology, economics, physics and mechanics and many more.
DDE are differential equations of the form:

$$y'(t) = f\left(t, y(t), y\left(t - \tau_1\left(t, y(t)\right)\right), y\left(t - \tau_2\left(t, y(t)\right)\right), ...\right),$$

where (τ) can be a constant, a function of time only or a function of time and state (of t and y itself). If $(\tau) < 0$ then the DDE is known as a retarded DDE, it relies on terms from the past. If $(\tau) > 0$ then it is advanced and relies on term in the future of the system. Many of the examples we found use a constant delay, sometimes because the exact nature of what causes the delay is not known precisely and is therefore estimated or varied and the effects studied. There are also Neutral DDE where the derivative is dependent on a previous derivative in the system (so the delay appears in a derivative term as well). There are also stochastic delays that rely on random variables. Although we will be looking a DDE with a stochastic term it will have a constant delay rather than the delay term being stochastic.

The DDE we will consider is the following linear DDE:

$$y'(t) = \lambda y(t-1) \tag{1}$$

3 Stability of delay differential equations

A solution to a DDE is said to be stable as t$\rightarrow \infty$ if given two positive numbers $t_0$and ε_t there exists a corresponding $\delta > 0$ such that for every continuous solution y(t)that satisfies

$$max_{t_0 \leq t \leq t_0+n} \left| y\left(t\right) - x\left(t\right) \right| \leq \delta$$

will also satisfy

$$max_{t_0 \leq t} \left| y\left(t\right) - x\left(t\right) \right| \leq \varepsilon$$

It is asymptotically stable if:

- It is stable
- For $t_0 \geq 0$ there is a δ such that every solution y(t) also satisifies

$$lim_{t \rightarrow \infty} \left| y\left(t\right) - x\left(t\right) \right| = 0$$

$y\left(t\right) \equiv 0$ is an equilibrium solution which will remain asymptotically stable if all roots remain asymptotically stable.
For $y'\left(t\right) = \lambda y\left(t-1\right)$ the asymptotic stability will depends on λ as shown in following section.

4 Characteristic equations and roots

For all continuous solutions to approach zero as $t \to \infty$, all roots of it's characteristic equation must have negative real parts.

As in the case for linear ODE's, for a linear DDE we want a solution of the form:

$$u(t) = e^{at}$$

Which gives:

$$u'(t) = ae^{at}$$

Replacing $y'(t)$ for ae^{at} in (1) we get:

$$ae^{at} = \lambda e^{a(t-1)}$$

$$ae^{at} = \lambda e^{at} e^{-a}$$

Cancelling e^{at} gives:

$$a = \lambda e^{-a}$$

This is the characteristic equations for (1).

For the system to be stable all the real roots of this characteristic equation need to be negative. For the above example this is the case for values of $\lambda > -\frac{\pi}{2}$, however, for values of $\lambda < -\frac{\pi}{2}$, the characteristic functions grow without bounds. The point at $\lambda = -\frac{\pi}{2}$ is the point where the behaviour of the system changes and it loses stability. This is a bifurcation point (see section 11).

5 Numerical solutions and theta methods for delay differential equations

Some ODE and DDE are not possible to solve analytically, in these cases we use numerical methods to solve them. Numerical methods involve discretizing the scheme into time steps and evaluating the system at each time step. There are two main types, linear multipstep methods and Runge-kutta methods. Linear multipstep methods use small increments in time to derive the next time step of the solution and continue in the same fashion to map out the entire solution. Runge-kutta methods split each step into intervals to create a higher order method but then discard these values and starts over for the next step. Within these methods there are implicit and explicit variations. Explicit methods calculate the next state entirely from the previous one, for example:

$$Y\left(t+\triangle t\right)=F\left(Y\left(t\right)\right)$$

Whereas implicit methods use a combination of the previous state and the new one:

$$Y\left(t+\triangle t\right)=G\left(Y\left(t\right),Y\left(t+\triangle t\right)\right)$$

To derive a numerical scheme we start by using Taylor expansion to derive a finite difference approximation for the differential $f'\left(x\right)$:

$$f'\left(x\right)=\frac{f\left(x+h\right)-f\left(x\right)}{h}$$

We then substitute this for the differential in our approximation. Applied to our delay differential equation this gives:

$$x'\left(t\right)=\lambda x\left(t-1\right) \tag{2}$$

$$\frac{u\left(t+h\right)-u\left(t\right)}{h}=\lambda x\left(t-1\right)$$

$$x\left(t+h\right)=x\left(t\right)+h\lambda x\left(t-1\right)$$

We then introduce $t=nh$ such that $Nh=1$, which splits each time interval into N steps of size h

This gives:

$$x\left(nh+h\right)=x\left(nh\right)+h\lambda x\left(nh-Nh\right)$$

$$x\left(\left(n+1\right)h\right)=x\left(nh\right)+h\lambda x\left(\left(n-N\right)h\right)$$

Let

$$x\left(nh\right)=x_n$$

$$x_{n+1}=x_n+h\lambda x_{n-N}$$

The following is the numerical scheme for the Euler-Theta methods:

$$x_{n+1}=x_n+h\left[\theta\lambda x_{n-N+1}+\left(1-\theta\right)\lambda x_{n-N}\right]$$

Here there are two instances of αx with a new parameter θ. The value of θ determines which of the Euler methods is implemented.
When $\theta=0$ we have the forward Euler method, where the next solution is derived entirely from the λx_{n-N} term.
When $\theta=1$ we have the backwards Euler method, where the next solution is derived entirely from the λx_{n-N+1} term.
When $\theta=\frac{1}{2}$ we have the trapezium method, where the solution is derived from a combination of both terms.

The Euler-theta methods can be represented as matrices:

$$\begin{bmatrix} x_{n+1} \\ x_n \\ x_{n-1} \\ \vdots \\ \vdots \\ x_{n-N+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & h\lambda\theta & h\lambda(1-\theta) \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots \\ \vdots & & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ x_{n-1} \\ x_{n-2} \\ \vdots \\ \vdots \\ x_{n-N} \end{bmatrix}$$

which is of the form:

$$x_{n+1} = Ax_n$$

Where A is called the companion matrix and each n step can be determined as follows:

$$x_{n+2} = A^2 x_n$$

$$x_N = A^N x_0$$

The matrix X_n contain the values for the previous time steps that are being used to calculate the current one, the 'history' of the system needed due to the delay term. The matrix X_{n+1} also contains X_n to X_{n-N+1} of them. The dimensions of these matrices will be determined by the number of the steps the interval has been split into (N). We use this property when we construct our code for the Euler-Maruyama scheme later.

Groovy code for the Euler-theta method for DDE:

```
import Jama.*
f = new File('ThetaOutput130-N10-T0.txt')

T = 1000
N = 10
h = 1/N
Theta = 0
def Infunc = {y -> y+0.5}
lambda = -1.30
Matrix Xn = new Matrix(N+1,1)

for (j in 0..N){
Xn.set(j,0,Infunc(j/N)*-1)
}

Matrix AMatrix = new Matrix(N+1,N+1)
AMatrix.set(0,0, 1)
AMatrix.set(0,N,h*lambda*(1-Theta))
AMatrix.set(0,(N-1),h*lambda*Theta)

for (k in 1..N){
AMatrix.set(k,k-1,1)
}
def Sln = []
Sln[0] = Xn.get(0,0)
```

```
for (i in 0..T-1){
Matrix Xn1 = AMatrix.times(Xn)
Sln[i] = (Xn1.get(0,0))
Xn = Xn1
}

for (j in 0..T-1){
f.append(Sln[j] + '\n')
}
```

The graphs in Figures 1 and 2 illustrate the approximate solutions given by the Euler-Theta method for (2) for $\theta = 0,\ 0.5,\ 1$. Figure 1 have $N = 4$ and with $\lambda = -1.30$ and $\lambda = -1.80$ these are the extreme ends of the values of λ we will be using later in this paper and show the extreme differences in the system. The graphs on the left show that the system is stable for values of $\lambda = -1.30$ with all solutions converging to 0, whereas the graphs on the right with $\lambda = -1.80$ show that the system is diverging and is no longer stable. As mentioned previously, we expect the value at which this change occurs to be approximately -1.57. Figure 2 illustrates the same thing but for $N = 10$ and shows the impact of increasing the number of steps used in the numerical approximation.

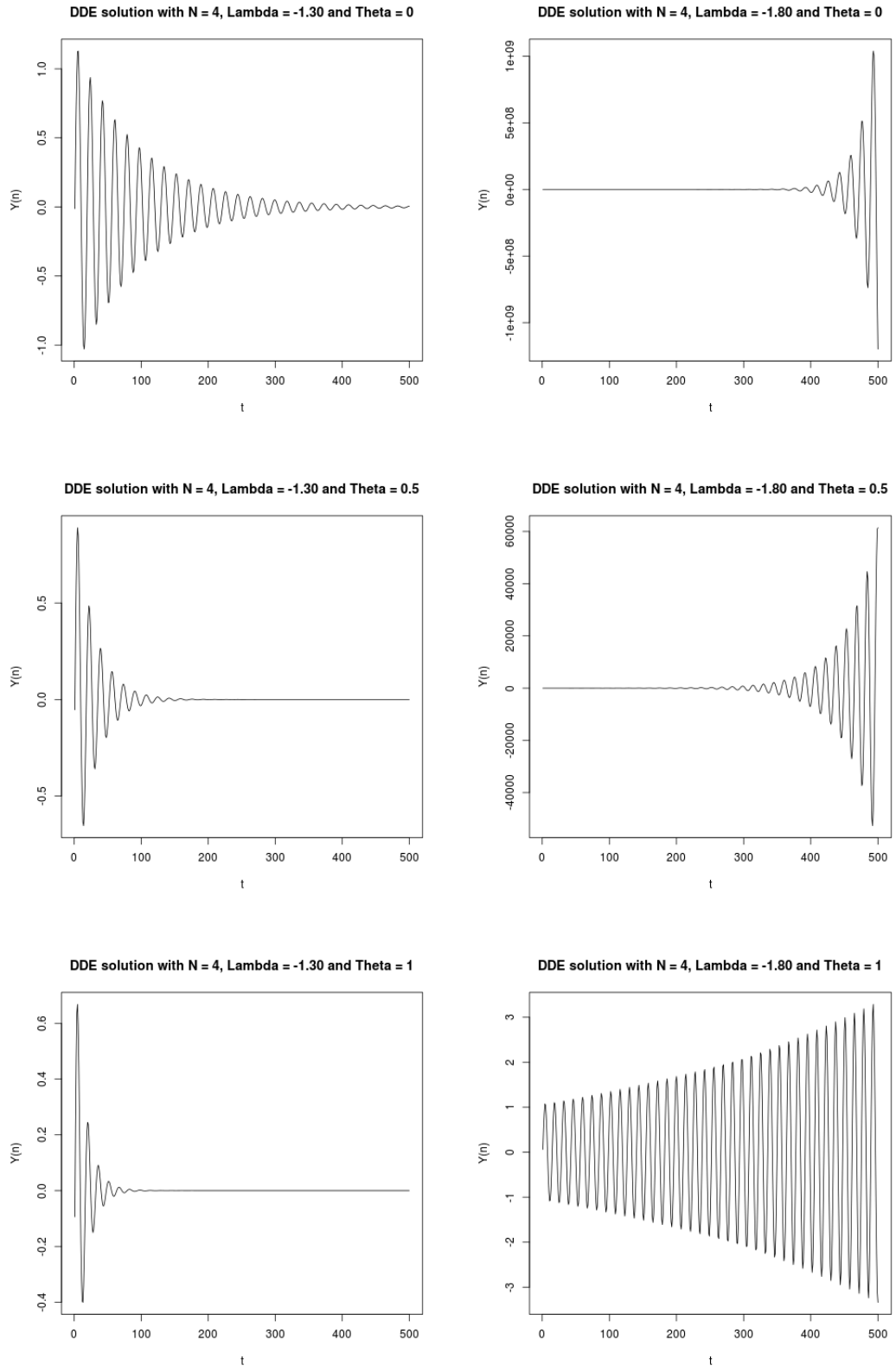
DDE solution with N = 4, Lambda = -1.30 and Theta = 0
DDE solution with N = 4, Lambda = -1.80 and Theta = 0
DDE solution with N = 4, Lambda = -1.30 and Theta = 0.5
DDE solution with N = 4, Lambda = -1.80 and Theta = 0.5
DDE solution with N = 4, Lambda = -1.30 and Theta = 1
DDE solution with N = 4, Lambda = -1.80 and Theta = 1
Y(n)
t

Figure 1:

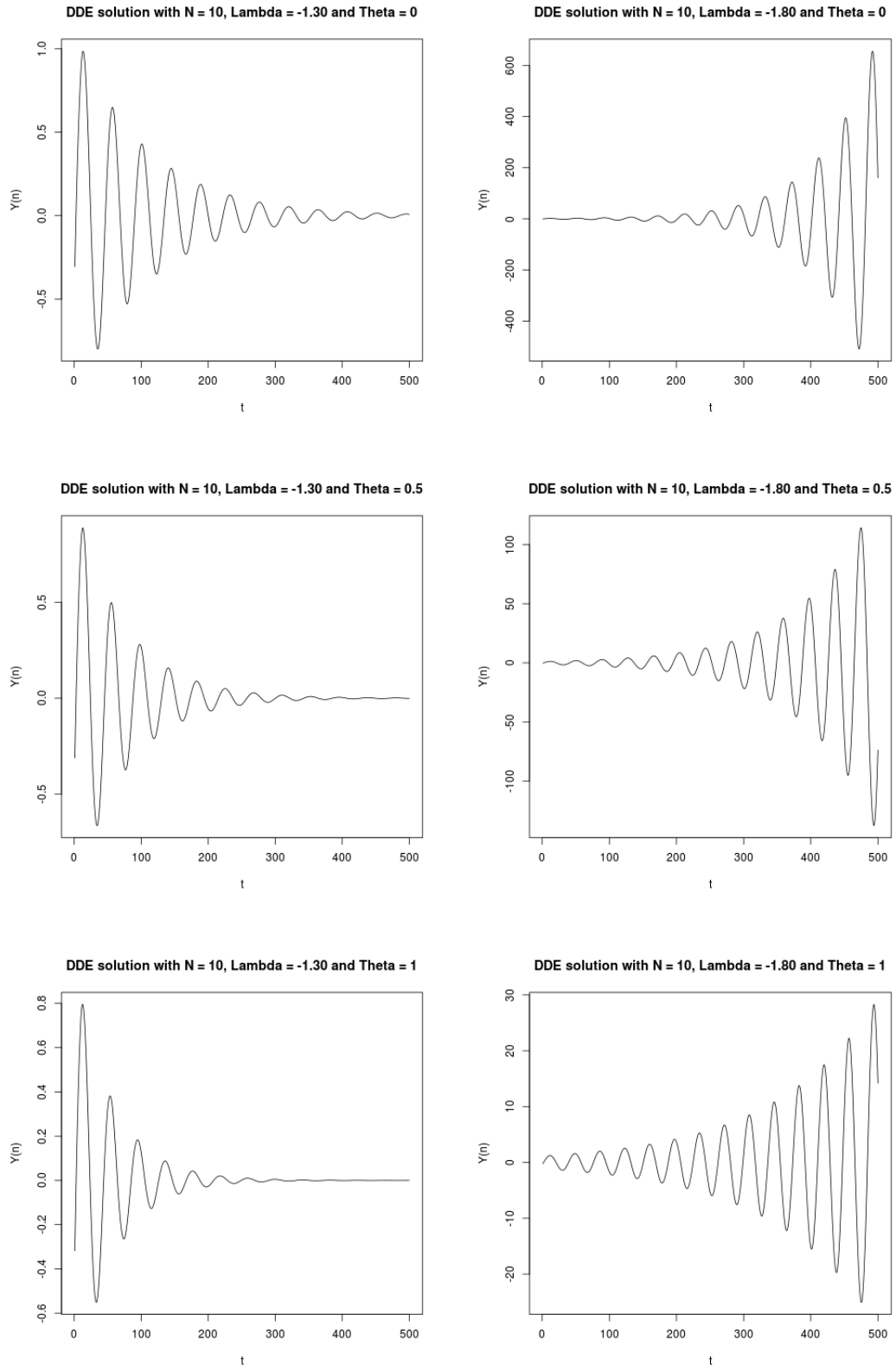
DDE solution with N = 10, Lambda = -1.30 and Theta = 0
DDE solution with N = 10, Lambda = -1.80 and Theta = 0
DDE solution with N = 10, Lambda = -1.30 and Theta = 0.5
DDE solution with N = 10, Lambda = -1.80 and Theta = 0.5
DDE solution with N = 10, Lambda = -1.30 and Theta = 1
DDE solution with N = 10, Lambda = -1.80 and Theta = 1
Y(n)
t

Figure 2:

6 Brownian motion

Some random events can have different probability distributions, making some outcomes more or less likely, it can consist of a form of noise or interference on a deterministic system simply muddling or masking the true event. Random variables in mathematical models crop up in finance, medicine, enigineering, game theory, the list is endless. One of the most prolific forms of random model that appears in most of the areas mentioned above is the Wiener process (otherwise known as Brownian motion). It is a normally distributed random variable, the discretised version (limited to values of +/-1) is often decribed as a simple random walk, or drunken walk (where each step the drunk makes is in an unknown direction, one step at a time).

Brownian motion was named after the biologist Robert Brown. He observed how particles of pollen suspended in water moved erratically in seemingly random and unpredictable patterns. Brown never managed to describe the behaviour mathematically, this has been acredited to Louis Bachelier who first defined Brownian motion as a stochastic process in 1900, closely followed by Einstein in 1905.

Brownian motion is a stochastic process which satifies the following properties:

- $W_0 = 0$, initial condition
- If $0 \leq s < t$ then $W_t - W_s \sim N(0, t-s)$, normally distributed with $mean = 0$ and standard deviation $t - s$
- Future changes are independent of past and present, if $0 \leq r \leq s < t$ then $W_t - W_s$ and W_r are independent
- The paths $t \to W_t$ are continuous with probability 1

Einstein and Bachelier did not prove these properties however, it was Norbert Weiner (1923) who did and also developed related mathematical theories. Because of this it has become convention to use the term Brownian motion with regards to the physical phenomenon and Wiener process for the mathematical model.

Brownian motion is a continuous function which is nowhere differentiable.

Using Taylor expansion as before to derive a finite difference approximation and taking the limit for the differential dW_t gives:

$$\lim_{h \to 0} \left(dW_t = \frac{\mid W_{t+h} - W_t \mid}{h} \right) = \lim_{h \to 0} \frac{c}{\sqrt{h}} = \infty$$

Which shows that the differential should not exist.
Re-arranging gives:

$$dW_{t+h} = W_{t+h} - W_t$$

7 Ito calculus

As stated above, one of the properties of Brownian motion is the fact that it is nowhere differentiable, so it will cause an otherwise easily solved system of simple differential equations to become complex. Developed by Kiyoshi Ito and published in 1942, Ito calculus is a method that enables the solutions of such systems and models.

In Lectures 3 and 4 of his lecture notes for Mathematical modelling as part of an Msc course in Financial Engineering, Raymond Brummelhuis [4] explains some of the key similarities and differences between classical and Ito calculus. Firstly, dt is restricted to being a positive differential. We also want to consider infinitesimally small time increments on the interval $[t, t+dt]$ where dt is interpreted as being a number so small that higher powers can be discarded.

$$dt \neq 0,\ (dt)^2 = (dt)^3 = \cdots = 0$$

Ito calculus uses the property that a normal random variable has mean 0 and variance dt and as stated previously, if $0 \leq s \leq t$ then $W_t - W_s \sim N(0, t-s)$

then with $h = dt$ we have:

$$E(|dW_t|) = c\sqrt{dt}$$

If dW_t is a differential of size $\sqrt{dt}$ then for very small dt it is bigger than itself, $\sqrt{dt} \geq dt$, Brummelhuis gives the numerical example $10^{-50} \geq 10^{-100}$.

It follows:

$$E\left((dW_t)^2\right) = dt$$

and

$$Var\left((dW_t)^2\right) = (dt)^2 = 0$$

This shows that dW_t is no longer a random variable but a value with no variance and equal to its mean.
To see how this works the following is an example of calculating the differential:

$$d\left(W_t^2\right) = \left(W_{t+dt}\right)^2 - W_t^2$$

$$= \left(W_t + dW_t\right)^2 - W_t^2$$

$$= \left(W_t^2 + 2W_t dW_t + \left(dW_t\right)^2\right) - W_t^2$$

$$= 2W_t dW_t + \left(dW_t\right)^2$$

$$= 2W_t dW_t + dt$$

Rather than fully derive Ito's Lemma for the purpose of this paper the following is a summary of some of the basic rules:

- $(dt)^3 dW_t = 0$
- $(dt)^2 dW_t = 0$
- $dt\left(dW_t\right)^2 = 0$
- $dt^2 = 0$
- $dtdW_t = 0$

Returning to Brownian motion and applying the concepts of Ito calculus as described we get the following:

$$df\left(W_t\right) = f\left(W_{t+dt}\right) - f\left(W_t\right)$$

since

$$W_{t+dt} = W_t + dW_t$$

Using Taylor expansion up to order $k = 2$ we have the following:

$$f\left(W_{t+dt}\right) = f\left(W_t\right) + f'\left(W_t\right) dW_t + \frac{1}{2} f''\left(W_t\right)\left(dW_t\right)^2 + O\left(\left(dW_t\right)^3\right)$$

Which when applying the basic rules of Ito calculus reduces to:

$$f\left(W_{t+dt}\right) = f\left(W_t\right) + f'\left(W_t\right) dW_t + \frac{1}{2} f''\left(W_t\right) dt$$

This is the method we employ to differentiate our stochastic term in our numerical approximation. Combining this with the Euler-Theta method already shown, this gives the Euler-Maruyama scheme.

We will return to Ito calculus in more detail and how it is used to derive the numerical scheme included in this paper when we discuss the Milstein scheme in section 10.

8 Euler-Maruyama for stochastic differential equations

The Euler-Maruyama method is essentially the Euler-Theta method for stochastic differential equations with Ito calculus applied to the stochastic term. It is often referred to as one of the easiest forms of numerical methods for stochastic differential equations ([7] p305).

We will first illustrate an example of the Euler-Maruyama method and then adapt this for our delay equation.

$$dY_t = (Y_t)\, dt + \mu\,(Y_t)\, dW_t$$

$$x_{n+1} = x_n + \theta h x_{n-N+1} + (1-\theta)\, x_{n-N} + \mu x_n \Delta W_n$$

which is:

$$x_{n+1} = x_n + h\left[\theta x_{n-N+1} + (1-\theta)\, x_{n-N}\right] + \mu x_n \Delta W_n$$

It can be seen here that when $\mu = 0$, the system reverts to the theta method.

Using an exact solution calculated from

$$X\,(t) = X\,(0) \exp^{\left(\left(\lambda - \frac{1}{2}\mu^2\right)t + \mu W(t)\right)}$$

The following code is a groovy translation of the code used in [5]:

Code for Euler-Maryama for SDE:

```
import Jama.*
import java.security.*

SecureRandom random = new SecureRandom()

T = 1
n = 2**8
X0 = 1
dt = T/n
```

```
def Gnums = [ ]
def dW = [ ]
def W = [ ]

for (i in 0..n){
dW[i] = Math.sqrt(dt)*random.nextGaussian()
}

lambda = 2
mu = 1
def f = {x -> x*lambda}
def g = {x -> x*mu}
N = 1
deltaT = N*dt
int steps = n/N
Matrix Xzeros = new Matrix(steps,1)
Xn = X0
Matrix Soln = new Matrix(steps,1)

for (k in 0..steps-1){
dWt = W[N+k]-W[k]
Xzeros.set(k,0, Xn + deltaT*f(Xn) + g(Xn)*dWt)
XOld = Xzeros.get(k,0) Soln.set(k,0, X0*Math.E**(((lambda-(0.5*mu**2))*k*dt)+(mu*W[k])))
}

def ApproxSol = [ ]
def ExactSol = [ ]
for (m in 0..steps-2){
ApproxSol[m] = Xzeros.get(m,0)
ExactSol[m] = Soln.get(m+1,0)
}

for (n in 0..steps-2){
println ApproxSol[n] + "\t" + ExactSol[n]
}
```

The graph in Figure 3 shows the Euler-Maruyama approximation plotted again the exact solution:

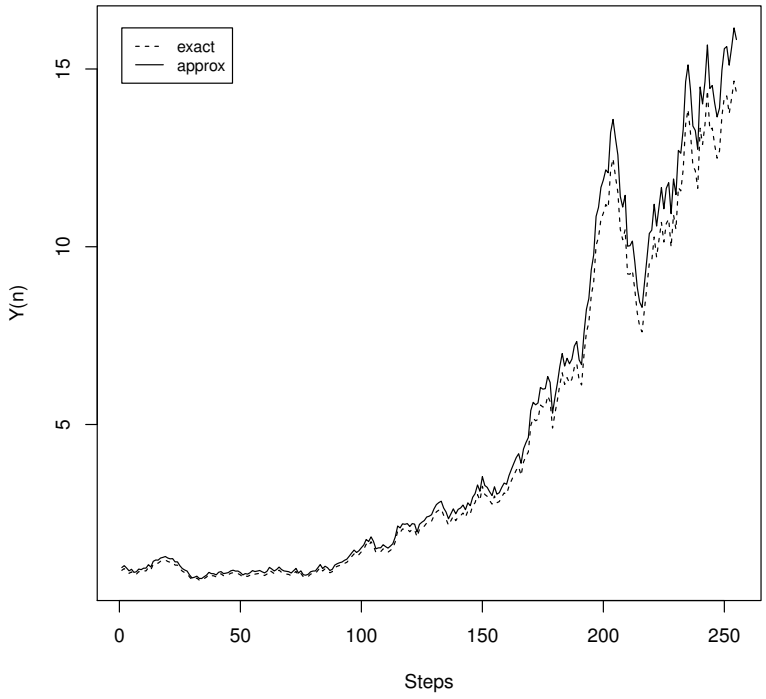

Figure 3:

Although the numerical methods performs relatively well when it comes to approximating the solution of the system there is a fair amount of deviation. The degree of how far the approximate solution differs from the exact solution (the error) or the accuracy is classified under the order of the method and is the topic of the next section.

9 Order and convergence of numerical methods

How accurately a numerical method represents the actual solution of a differential equation is known as the convergence of the system and is usually refered to as being either strong or weak. Strong convergence refers to the step by step fit (pathwise fit) of the approximation and is assessed by means of the total absolute error between the approximation and the exact solution of the system. It is a measure of how closely the numerical method maps each point along the entire path. Weak convergence is a measure at moments only. As we are considering the approximations of entire trajectories of solutions in this paper, we are primarily interested in the order of the strong convergence of the numerical methods used.

As can be seen in the graphs in section 5, strong convergence can be affected by the step size chosen for the model. For the Euler schemes it can be shown using the log of error against the log of the step size, that the absolute error is proportional to $\sqrt{h}$ where h is the step size.

We state the definition given in [2]:
Strong convergence:
A method is said to have strong convergence of order γ if there exists a constant C such that, for a sufficiently small step length h, $E|Y - Y(t_n)| \leq Ch^{\gamma}$ where $Y(t_n)$is the exact value of the solution at gridpoint $t_n \epsilon [0, 1]$.

Weak convergence uses mean error and is also proportional to $\sqrt{h}$.

Again we state the definition given in [2]:
Weak convergence:
A method is said to have weak convergence of order γ if there exists a constant C such that, for sufficiently small step length h, $|E(Y_n) - E(Y(t_n))| \leq Ch^{\gamma}$.

The Euler-Maruyama scheme we have already introduced is known to be strongly convergent of order 0.5 and weakly convergent of order 1.

10 Milstein method

The choice of the numerical method used can be shown to influence the skew from the theoretic value of the root of the characteristic equation. Wulf [3] indicated that although the numerical methods retain the information they deviate depending on the order of the method used.

"Numerical methods for DDE retain qualitative features for DDE undergoing Hopf bifurcations.
For general linear $\theta - methods$ - found higher order methods behaved better in that they approximate the bifurcation points to a higher order."

The Milstein numerical method is known to be both strongly and weakly convergent of order 1.
Milstein in his paper [11] showed that schemes such as the Euler-Maruyama have a mean square deviation of $E_{t_0,x_0}\left(X\left(t_0+T\right)\right)-\left(X\left(\bar{t_0}+T\right)\right)^2$ which is a value of $O\left(h\right)$, he proposed a method that by using only W_{k+1} and additional random variables at each stage gave an accuracy of $O\left(h^2\right)$

The Milstein scheme simply has an extra term that was not used in the Euler-Maruyama scheme. To explain where this comes from we need to return to Ito calculus for a moment. Ito's formula, which we did not include earlier, states that for a given function that is twice differentiable we can write:

$$f\left(X_s\right)=f\left(X_{t_i}\right)+\int_{t_i}^{s}\left(f'\left(X_u\right)a\left(X_u\right)+\frac{1}{2}f''\left(X_u\right)b\left(X_u\right)^2\right)du+\int_{t_i}^{s}f'\left(X_u\right)b\left(X_u\right)dW_u$$

If we apply this to the following SDE as described in [38]

$$dX_t=a\left(X_t\right)dt+b\left(X_t\right)dW_t$$

We get:

$$X_{t+1} = X_t + \int_{t_i}^{t_{i+1}} \left(a\left(X_{t_i}\right) + \int_{t_i}^{s} \left(a'\left(X_u\right) a\left(X_u\right) + \frac{1}{2} a''\left(X_u\right) b^2\left(X_u\right) \right) du + \int_{t_i}^{s} a'\left(X_u\right) b\left(X_u\right) dW_u \right) ds$$

$$+ \int_{t_i}^{t_{i+1}} \left(b\left(X_{t_i}\right) + \int_{t_i}^{s} \left(b'\left(X_u\right) a\left(X_u\right) + \frac{1}{2} b''\left(X_u\right) b^2\left(X_u\right) \right) du + \int_{t_i}^{s} b'\left(X_u\right) b\left(X_u\right) dW_u \right) dW_s$$

We then discretise the time intervals and using the following properties and discard terms of order higher than 1:

- $\delta t \cdot \delta t = \mathcal{O}\left((\delta t)^2\right)$
- $\Delta W \cdot \delta t = \mathcal{O}\left((\delta t)^{\frac{3}{2}}\right)$
- $\Delta W \cdot \Delta W = \mathcal{O}\left(\delta t\right)$

We get:

$$X_{t_{i+1}} \approx X_{t_i} + \int_{t_i}^{t_{i+1}} a\left(X_{t_i}\right) ds + \int_{t_i}^{t_{i+1}} \left(b'\left(X_u\right) b\left(X_u\right) dW_u \right) dW_s$$

$$\approx X_{t_i} + a\left(X_{t_i}\right) \delta t + b\left(X_{t_i}\right) \Delta W_i + \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} b'\left(X_u\right) b\left(X_u\right) dW_u dW_s$$

$$\approx X_{t_i} + a\left(X_{t_i}\right) \delta t + b\left(X_{t_i}\right) \Delta W_i + b'\left(X_{t_i}\right) b\left(X_{t_i}\right) \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} dW_u dW_s$$

$$\approx X_{t_i} + a\left(X_{t_i}\right) \delta t + b\left(X_{t_i}\right) \Delta W_i + b'\left(X_{t_i}\right) b\left(X_{t_i}\right) \frac{1}{2} \left((\Delta W)^2 - \delta t \right)$$

In our SDDE $a = 1$, $b = \mu$ and $\delta t = h$, substituting for these we get:

$$x_{n+1} = x_n + (x_n)\, h + \mu\,(x_n)\,\Delta W_n + \frac{1}{2}\mu'\,(x_n)\,\mu\,(x_n)\left((\Delta W_n)^2 - h\right)$$

This does not yet include our delay term, adding this in the same way as we did for the Euler-Maruyama scheme we get:

$$x_{n+1} = x_n + h\left[\theta\lambda x_{n-N+1} + (1-\theta)\,\lambda x_{n-N}\right] + \mu x_n dW_t + \frac{1}{2}\mu^2 x_n\left(dW_t^2 - h\right)$$

11 Bifurcation & Lyapunov exponents

When systems become unstable they display chaotic behaviours. The point at which these occur are described as various types of bifurcation points. Some local examples of these include Hopf bifuractions, pitchfork, saddle-node, period doubling and transcritical.

These points are easy to see when results are graphed and there has been much work surrounding the stability regions of systems. However it is not always an easy or accurate method. Another alternative approach is with Lyapunov exponentials. Lyapunov exponents are a measure of how solution trajectories diverge over time. This method will detect when a system converges to a fixed point or displays periodic cycling. We are interested in using them here to detect when a change in the system parameters results in instability.

When a Lyapunov exponent is positive it is an indicator of a chaotic system. In stochastic delay differential equations (SDDE) there will be an infinte number of Lyapunov exponents describing the system. We are only concerned with the largest of the Lyapunov exponents as this one indicates the largest divergence and therefore the greatest amount of instability in the system.

$$S = \sup_{T-\varepsilon,T} \left(|Y(t)| \right)$$

$$\Lambda = \lim_{t\to\infty} \sup E\left(\frac{1}{t} \log |Y(t)| \right)$$

One of the downfalls of using Lyapunov exponents to detect the bifurcation points in stochastic systems is that they are used to detect deterministic chaos and and can lead to 'false positives' in systems where there is too much noise or randomness . Jonathan Dingwell [12] among many other authors [17,19] claims that finding a positive Lyapunov exponent by itself should not be taken as proof of chaos, Franca and Savi [19]also showed that some algorithms (in particular Wolf's algorithm [9]) was especially sensitive to noise. As we are using them to determine an approximation for bifurcations points in a stochastic system, and although Norton investigated the impact of varying the level of noise, there will be a point where you are guaranteed a positive Lyapunov exponent due to the noise rather than the underlying system.

Yonemoto and Yanagawa [14] claim that
"Lyapunov exponents are reliable if the data are abundant, if the measurement error is near 0 and if the data really come from a deterministic system. However, with limited data or a system subject to non-negligable stochastic pertubations, it is well known that the estimates

may be incorrect or ambiguous."

Dennis, Desharnais, Cushing, Henson and Constantino [17] showed it was possible for the stochatic Lyapunov exponents to be positive when the Lyapunov exponents of the underlying deterministic model is negative and vice versa. They also claimed it "should not be viewed as a hallmark of chaos" and Argyris and Andreadis [15] found that the largest Lyapunov exponent keeps growing under the influence of noise.

There are plenty of examples in research such as Dingwell [12], Lui, Dai, Li, Gong [13] and Kostelich and Yorke [18] where methods are suggested for either removing the noise from data or making a 'correction' for the noise before calculating the Lyapunov exponents. As it appears to be such a minefield and we do not want to introduce any factors that may not be due to the numerical method or step size we have chosen but due to the performance of the Lyapunov exponents as an indicator when we increase or decrease noise levels, we have decided to fix μ at 0.1 throughout this paper.

12 Methodology

Wulf [3] proved that numerical methods retain information about bifurcation points and perturbations are $O(h^n)$ where n is the order of the numerical scheme.

There are several different DDE's solvers available. Many use a Runge-Kutta method. One of the first things to consider is whether the method used in a DDE solver is appropriate for the model it is to be applied to. Some are general whereas some can be specifically for a certain type of DDE. Other considerations include the amount of memory needed to calculated the solutions due to the amount of history of the sytem that needs to be stored to calculate the solution. Also, the way discontinuities are tracked and handled. Some numerical solvers can create 'ghost' solutions where theoretically they don't exist.

However, for this investigation we have chosen to build a model using the Java scripting language Groovy. This allowed us to understand fully the structure of the numerical process and how each one can be adapted and extended to allow further modifications to the model. The code is saved as a groovy file and can be run from the command line.

For example, from starting with a basic $\theta - method$ for simple deterministic DDE's we were able to add the stochastic term which turned the model into an Euler-Maryama type model that allows delays and then simply needed to add an extra term again to create an Milstein version of the same thing (seen later in section 14).

Combining them is simply adding the brownian motion term (with reverence to the x_n term from the matrix not the x_{n+1}).
which gives:

$$x_{n+1} = x_n + h\left[\theta\lambda x_{n-N+1} + (1-\theta)\lambda x_{n-N}\right] + \mu x_n dW_t$$

The stochastic terms were stored in text files so that the same values could be used for all calculations, this way the only influencing factor should be the changes to the models and not due to random skews between different runs of random values. It would also allow for different sources of random numbers to be used, so instead of relying on just a pseudo RNG we could also test the results with 'true' random numbers (ie, from a source such as Random.org).

As previously stated, Lyapunov exponents become ineffective as an indicator of chaos/ stability when the level of noise in the system goes beyond a certain level. To avoid introducing any false results due to this and to enable us to establish a relational equation for the Lyapunov

exponent in terms of h and λ,we intend to keep the stochastic term μ fixed at 0.1 throughout.

In the following code we use the fact that the matrices for calculating the delay terms store the history and we reference the x_n term using Xn1.get(1,0) and Xn1.get(0,0) for the x_{n+1} term.

Combined code for SDDE:

```
import Jama.*

f = new File('OutputTest1-50.txt')

for (r in 1..250){
def LE = 0
epsilon = 50
T = 5000
N = 10
h = 1/N
Theta = 0
def Infunc = {y -> y+0.5}
lambda = -1.50
mu = 0.1
def Gnums = [ ] new File(r + 'Gaussian.txt').eachLine{ Gnums.add(it.toFloat()) }
def dW = [ ]
def W = [ ]

for (i in 0..T*N){
dW[i] = Math.sqrt(h)*Gnums[i]
W[i] = dW.sum()
}
Matrix Xn = new Matrix(N+1,1)

for (j in 0..N){
Xn.set(j,0,Infunc(j/N)*-1)
}
Matrix AMatrix = new Matrix(N+1,N+1)
AMatrix.set(0,0, 1)
AMatrix.set(0,N,h*lambda*(1-Theta))
AMatrix.set(0,(N-1),h*lambda*Theta)

for (k in 1..N){
AMatrix.set(k,k-1,1)
}
def SIn = [ ]
SIn[0] = Xn.get(0,0)
```

```
for (i in 0..T-1){
dWt = W[i+N]-W[i] //Delta Wn
Matrix Xn1 = AMatrix.times(Xn)
Sln[i] = (Xn1.get(0,0) + mu * dWt * Xn1.get(1, 0))
                          extra stochastic term
Xn = Xn1
}
def LEtemp = [ ]

for (n in 0..epsilon-1){
LEtemp[n] = Math.abs(Sln[T-epsilon+n])
}
```

The graphs in Figure 4 are the solutions for our first simulations with $\lambda = -1.30$ and $\lambda = -1.80$ which show the same behaviours as the non-stochastic DDE:

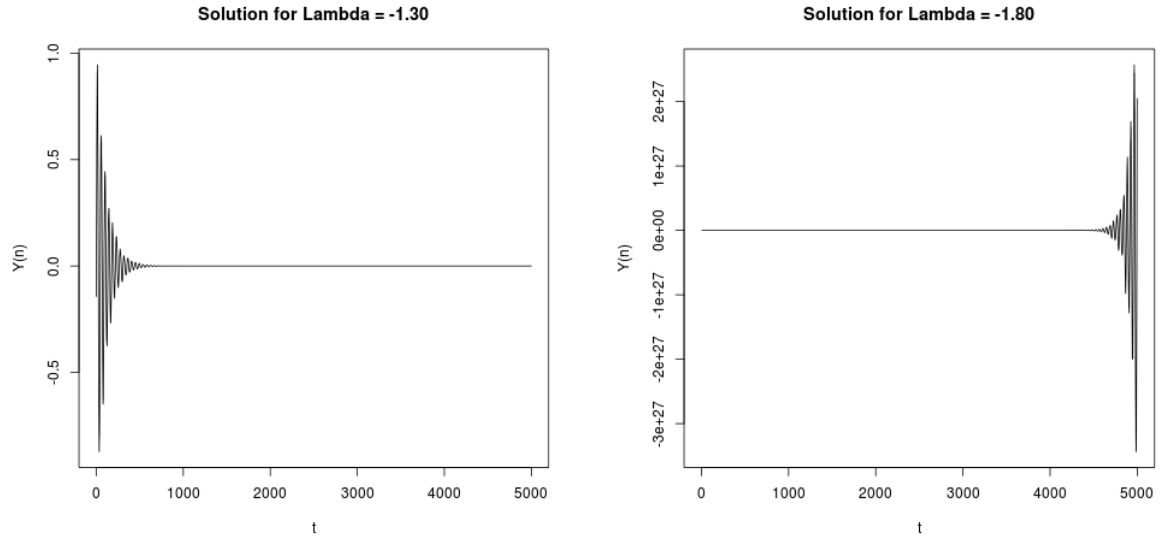

Figure 4:

13 Verification of Norton's results

We followed Norton's methodology (as stated in [1]) initially using $T = 5000$ and $\varepsilon = 5$ and for all values of λ, the Lyapunov exponents for 500 trajactories were calculated. The following tables are the original tabulated results published in [1] with our results for $\theta = 0$, $h = 0.1$, $\mu = 0.1$. All our results had p-values approaching zero for a KS test.

Norton's results:

λ	min	mean	max	st dev
-1.80	0. 125696	0.127203	0.129467	0.000636
-1.78	0.117992	0.119548	0.121777	0.000577
-1.76	0.110093	0.111871	0.113710	0.000607
-1.74	0.102413	0.104070	0.105727	0.000623
-1.72	0. 094544	0.096156	0.097865	0.000572
-1.70	0.086109	0.088214	0.090290	0.000607
-1.68	0.078722	0.080134	0.081711	0.000559
-1.66	0.068938	0.071973	0.073805	0.000628
-1.64	0.061837	0.063713	0.065542	0.000618
-1.62	0.053654	0.055401	0.057312	0.000611
-1.60	0.045156	0.046968	0.048924	0.000636
-1.58	0.036722	0.038417	0.040472	0.000639
-1.56	0.027892	0.029741	0.032026	0.000658
-1.54	0.018852	0.021045	0.022701	0.000633
-1.52	0.010178	0.012110	0.014359	0.000628
-1.50	0.001550	0.003210	0.005319	0.000643
-1.48	-0.007740	-0.005871	-0.003872	0.000643
-1.46	-0.016888	-0.015106	-0.013391	0.000633
-1.44	-0.026084	-0.024429	-0.022442	0.000663
-1.42	-0.035916	-0.033894	-0.031982	0.000686
-1.40	-0.045154	-0.043385	-0.041458	0.000654
-1.38	-0.054813	-0.053148	-0.051360	0.000642
-1.36	-0.065142	-0.062943	-0.060652	0.000658
-1.34	-0.74742	-0.072934	-0.070458	0.000684
-1.32	-0.085634	-0.083084	-0.081031	0.000650
-1.30	-0.095153	-0.093354	-0.090911	0.000666

Our results:

λ	min	mean	max	st dev
-1.80	0.126142	0.126895	0.127347	0.000182
-1.78	0.119004	0.119499	0.119882	0.000169
-1.76	0.111195	0.111720	0.112142	0.000166
-1.74	0.103230	0.103803	0.104270	0.000170
-1.72	0.095429	0.095834	0.096316	0.000157
-1.70	0.087571	0.088098	0.088506	0.000164
-1.68	0.079516	0.079953	0.080373	0.000157
-1.66	0.071215	0.071728	0.07219	0.000160
-1.64	0.062908	0.063433	0.063914	0.000166
-1.62	0.054741	0.055152	0.055590	0.000134
-1.60	0.046339	0.046768	0.047153	0.000144
-1.58	0.037815	0.038189	0.038555	0.000138
-1.56	0.029171	0.029522	0.029956	0.000134
-1.54	0.020272	0.020755	0.021207	0.000144
-1.52	0.011328	0.011895	0.012302	0.000150
-1.50	0.002457	0.002932	0.003340	0.000148
-1.48	-0.023987	-0.011764	-0.010222	0.000897
-1.46	-0.019375	-0.019337	-0.019300	0.000012
-1.44	-0.027535	-0.027202	-0.026907	0.000099
-1.42	-0.036117	-0.036539	-0.035755	0.000123
-1.40	-0.046005	-0.045570	-0.045199	0.000126
-1.38	-0.055918	-0.055536	-0.055204	0.000112
-1.36	-0.066500	-0.066345	-0.066197	0.000048
-1.34	-0.092474	-0.081519	-0.078252	0.002101
-1.32	-0.086639	-0.085429	-0.084636	0.000301
-1.30	-0.095080	-0.094164	-0.093505	0.000241

Our results differ slightly from those presented in the paper, however apart from the obvious reason of having different Gaussian numbers, this could be down to having used groovy and

java instead of Matlab as they use different algrithms for scaling the uniform random numbers from the RNG to Gaussian distributed numbers, Matlab changed from using a Polar (double Box-Muller) method to a Ziggurat algorithm from versions 5 and up whereas Java uses the Polar algorithm, or possibly just advances and changes in computer processing power since Norton's original work was conducted.

We have lower values overall and a much tighter standard deviation. We achieved the comparable results from 250 trials with T=5000 instead of 500 whilst maintaining a much smaller standard deviation. For this reason and to save a considerable amount of time needed in runnning the simulations, we base the rest of our results on just 250 trials as we are attempting to simply verify Stewart's results before investigating whether the claims hold when using a higher order method, in this case the Milstein method.

Having generated values for the Lyapunov exponent for each value of λ shown in the table above, we used R to perform a linear regression to determine an equation for the average Lyapunov exponent (L_{mean}) in terms of λ.The following are the equations we derived for $\theta = 0,\ 0.5,\ 1$ for 250 trials:

For $\theta = 0$:

$$L_{mean} = -0.448422\lambda - 0.674168$$

$$L_0 = -1.506899$$

For $\theta = 0.5$:

$$L_{mean} = -0.464863\lambda - 0.735214$$

$$L_0 = -1.581571$$

For $\theta = 1$:

$$L_{mean} = -0.483876\lambda - 0.801878$$

$$L_0 = -1.657197$$

Although the values for λ and the intersect in the linear regressions deviate from those published in [2], the values for L_0 are comparable. Norton's values for L_0 were $-1.4942,\ -1.5741,\ -1.652$ for $\theta = 0,\ 0.5,\ 1$ respectively.

Although we know that the actual value of the bifurcation point of the deterministic equation is -pi/2 (from the characteristic equation in section 2), this will be different when applying the Lyapunov exponents to a numerical scheme which is an approximation. We determined that the Lyapunov exponents for the deterministic equations and the bifurcation values were -1.499821 for $\theta = 0$, -1.579603 for $\theta = 0.5$, and -1.654062 for $\theta = 1$. Although the value for $\theta = 0.5$ is close to the actual value it is still not exact.
This gives us a reference for the accuracy of the results of Milstein scheme and is in keeping with Wulf's findings [3].

14 Comparison of Milstein results

Using the same random numbers for the stocastic term that we used for the Euler-Maruyama method we then conducted the same calculation and methodolgy using a Milstein scheme.

Code for Milstein:

```
import Jama.*

f = new File('OutputTest1-50.txt')

for (r in 1..250){
def LE = 0
epsilon = 50
T = 5000
N = 10
h = 1/N
Theta = 0
def Infunc = {y -> y+0.5}
lambda = -1.50
mu = 0.1
def Gnums = [ ] new File(r + 'Gaussian.txt').eachLine{ Gnums.add(it.toFloat()) }
def dW = [ ]
def W = [ ]

for (i in 0..T*N){
//dW[i] = Math.sqrt(h)*random.nextGaussian()
dW[i] = Math.sqrt(h)*Gnums[i]
W[i] = dW.sum()
}
Matrix Xn = new Matrix(N+1,1)

for (j in 0..N){
Xn.set(j,0,Infunc(j/N)*-1)
}
Matrix AMatrix = new Matrix(N+1,N+1)
AMatrix.set(0,0, 1)
AMatrix.set(0,N,h*lambda*(1-Theta))
AMatrix.set(0,(N-1),h*lambda*Theta)

for (k in 1..N){
AMatrix.set(k,k-1,1)
}
def SIn = [ ]
```

```
Sln[0] = Xn.get(0,0)

for (i in 0..T-1){
dWt = W[i+N]-W[i]
Matrix Xn1 = AMatrix.times(Xn)
Sln[i] = (Xn1.get(0,0) + mu*dWt*Xn1.get(1,0))+(0.5 * (mu * *2) * Xn1.get(1, 0) * ((dWt * *2) − h)))
                                              extra term for Milstein
Xn = Xn1
}
def LEtemp = [ ]

for (n in 0..epsilon-1){
LEtemp[n] = Math.abs(Sln[T-epsilon+n])
}
LE = N*(Math.log(LEtemp.max()))/T
println LE
}
```

Computational considerations

We used a Dell R510 server with 8 virtual CPU's and 32.768 GB memory running 64 bit Ubuntu Linux. When calculating the approximations for 20 steps, the ones with the largest computational demand, 12 instances at a time took approximately 3 days to complete for the Euler-Maruyama scheme. There was no difference seen in the CPU capacity or the completion time needed when running the same for the Milstein scheme. The addition term in the calculation had little to no impact on the overall computations and the complexity is due solely to the delay term that is a component of both schemes. No advantage or disadvantage could be discerned between either of the two schemes.

The following results are our findings from the Euler-Maruyama for 250 trials alongside the results from the Milstein method, also for 250 trials with $h = 10$ and $\mu = 0.1$, we have included the tables for $\theta = 0.5$ and $\theta = 1$ to illustrate that the results are not isolated to just $\theta = 0$:

Tables of results for $\theta = 0$

Euler-Maruyama results:

λ	min	mean	max	st dev
-1.80	0.126142	0.126911	0.127325	0.000178
-1.78	0.119026	0.119503	0.119881	0.000158
-1.76	0.112114	0.111726	0.112114	0.000160
-1.74	0.103342	0.103814	0.104270	0.000170
-1.72	0.095459	0.095840	0.096313	0.000159
-1.70	0.087695	0.088103	0.088506	0.000153
-1.68	0.079522	0.079961	0.080373	0.000149
-1.66	0.071273	0.071741	0.072194	0.000160
-1.64	0.062908	0.063446	0.063887	0.000168
-1.62	0.054750	0.055147	0.055507	0.000133
-1.60	0.046342	0.046769	0.047153	0.000139
-1.58	0.037815	0.038195	0.038554	0.000134
-1.56	0.029171	0.029533	0.029956	0.000136
-1.54	0.020272	0.020769	0.021207	0.000141
-1.52	0.011488	0.011908	0.012302	0.000150
-1.50	0.002943	0.002474	0.003338	0.000154
-1.48	-0.023987	-0.011839	-0.01036	0.001071
-1.46	-0.019369	-0.019336	-0.01930	0.000012
-1.44	-0.027535	-0.027209	-0.026943	0.000102
-1.42	-0.036539	-0.036125	-0.035798	0.000127
-1.40	-0.046005	-0.045579	-0.045244	0.000131
-1.38	-0.055918	-0.055544	-0.055244	0.000116
-1.36	-0.06650	-0.066349	-0.066216	0.000049
-1.34	-0.09042	-0.081677	-0.078459	0.002204
-1.32	-0.086639	-0.085451	-0.084722	0.000313
-1.30	-0.09508	-0.094182	-0.093579	0.000251

Milstein results:

λ	min	mean	max	st dev
-1.80	0.126258	0.126921	0.127383	0.000184
-1.78	0.119025	0.119510	0.119918	0.000159
-1.76	0.111254	0.111734	0.112162	0.000166
-1.74	0.103377	0.103824	0.104340	0.000178
-1.72	0.0954612	0.095851	0.096389	0.000169
-1.70	0.087710	0.088111	0.088557	0.000157
-1.68	0.079521	0.079971	0.0804312	0.000157
-1.66	0.071308	0.071752	0.072265	0.000169
-1.64	0.062920	0.063459	0.063960	0.000180
-1.62	0.054778	0.055158	0.055585	0.000142
-1.60	0.046371	0.046779	0.047206	0.000148
-1.58	0.037831	0.038208	0.038625	0.000144
-1.56	0.029176	0.029547	0.030063	0.000147
-1.54	0.020286	0.020782	0.021313	0.000154
-1.52	0.011499	0.011922	0.012413	0.000164
-1.50	0.002491	0.002957	0.003413	0.000169
-1.48	-0.006498	-0.006115	-0.005667	0.000160
-1.46	-0.015696	-0.015302	-0.014890	0.000155
-1.44	-0.024971	-0.024597	-0.024213	0.000155
-1.42	-0.034436	-0.034015	-0.033604	0.000158
-1.40	-0.043964	-0.043559	-0.043136	0.000160
-1.38	-0.053641	-0.053234	-0.052810	0.000162
-1.36	-0.063466	-0.063051	-0.062659	0.000164
-1.34	-0.073420	-0.073011	-0.072580	0.000174
-1.32	-0.083568	-0.083134	-0.082605	0.000185
-1.30	-0.093835	-0.093427	-0.092780	0.000187

Tables of results for $\theta = 0.5$

Euler-Maruyama results:

λ	min	mean	max	st dev
-1.80	0.095251	0.095734	0.096185	0.000174
-1.78	0.087585	0.087972	0.088348	0.000151
-1.76	0.079396	0.079855	0.080233	0.000154
-1.74	0.071197	0.071656	0.072118	0.000154
-1.72	0.062782	0.063380	0.063772	0.000160
-1.70	0.054728	0.055035	0.055463	0.000143
-1.68	0.046319	0.046731	0.047099	0.000138
-1.66	0.037779	0.038192	0.038551	0.000137
-1.64	0.029255	0.029573	0.029956	0.000130
-1.62	0.020432	0.020861	0.021299	0.000132
-1.60	0.011696	0.012047	0.012472	0.000133
-1.58	0.002711	0.003128	0.003604	0.000146
-1.56	-0.007915	-0.007009	-0.006402	0.000252
-1.54	-0.017825	-0.016794	-0.016131	0.000280
-1.52	-0.028054	-0.026813	-0.026062	0.000324
-1.50	-0.038476	-0.036908	-0.036043	0.000386
-1.48	-0.048482	-0.046693	-0.045762	0.000424
-1.46	-0.057415	-0.055860	-0.054999	0.000384
-1.44	-0.065926	-0.064705	-0.063962	0.000320
-1.42	-0.074643	-0.073632	-0.072977	0.000276
-1.40	-0.083711	-0.082822	-0.082223	0.000249
-1.38	-0.093156	-0.092345	-0.091786	0.000230
-1.36	-0.103021	-0.102271	-0.101744	0.000215
-1.34	-0.113415	-0.112730	-0.112238	0.000199
-1.32	-0.124671	-0.124103	-0.123677	0.000169

Milstein results:

λ	min	mean	max	st dev
-1.80	0.095250	0.095745	0.096259	0.000184
-1.78	0.087584	0.087980	0.088399	0.000155
-1.76	0.079408	0.079864	0.080272	0.000162
-1.74	0.071209	0.071667	0.072188	0.000162
-1.72	0.062796	0.063392	0.063831	0.000171
-1.70	0.054731	0.055048	0.055538	0.000156
-1.68	0.046349	0.046741	0.047150	0.000146
-1.66	0.037805	0.038204	0.038633	0.000147
-1.64	0.029256	0.029587	0.030015	0.000142
-1.62	0.020433	0.020876	0.021407	0.000145
-1.60	0.011695	0.012061	0.012576	0.000147
-1.58	0.002726	0.003142	0.003713	0.000161
-1.56	-0.006238	-0.005870	-0.005418	0.000166
-1.54	-0.015451	-0.014988	-0.014522	0.000169
-1.52	-0.024688	-0.024212	-0.023748	0.000167
-1.50	-0.034001	-0.033547	-0.33106	0.000165
-1.48	-0.043427	-0.043002	-0.042565	0.000166
-1.46	-0.053016	-0.052582	-0.052118	0.000171
-1.44	-0.062758	-0.062292	-0.061785	0.000177
-1.42	-0.072551	-0.072138	-0.071616	0.000182
-1.40	-0.082551	-0.082129	-0.081530	0.000181
-1.38	-0.092583	-0.092181	-0.091583	0.000168
-1.36	-0.102691	-0.102238	-0.101680	0.000192
-1.34	-0.113023	-0.112584	-0.112093	0.000184
-1.32	-0.123647	-0.123183	-0.122721	0.000179

Tables of results for $\theta = 1$

Euler-Maruyama results:

λ	min	mean	max	st dev
-1.80	0.064573	0.064949	0.065295	0.000137
-1.78	0.056172	0.056583	0.056968	0.000141
-1.76	0.047682	0.048057	0.048437	0.000142
-1.74	0.039080	0.039462	0.039893	0.000138
-1.72	0.030402	0.030779	0.031205	0.000138
-1.70	0.021577	0.022003	0.022449	0.000146
-1.68	0.012725	0.013127	0.013542	0.000150
-1.66	0.003843	0.004182	0.004579	0.000141
-1.64	-0.009440	-0.007992	-0.007166	0.000364
-1.62	-0.030953	-0.023219	-0.019816	0.002107
-1.60	-0027075	-0.027055	-0.027033	0.000007
-1.58	-0.035622	-0.035363	-0.035145	0.000083
-1.56	-0.044819	-0.044488	-0.044219	0.000104
-1.54	-0.054453	-0.054137	-0.053878	0.000099
-1.52	-0.064584	-0.064395	-0.064233	0.000061
-1.50	-0.076835	-0.076208	-0.075685	0.000200
-1.48	-0.087604	-0.085308	-0.084255	0.000499
-1.46	-0.094143	-0.093066	-0.092382	0.000290
-1.44	-0.103035	-0.102156	-0.101563	0.000246
-1.42	-0.112688	-0.111904	-0.111357	0.000224
-1.40	-0.12976	-0.122266	-0.121759	0.000206
-1.38	-0.134095	-0.133488	-0.133039	0.000180
-1.36	-0.147764	-0.147661	-0.147555	0.000037
-1.34	-0.156505	-0.155577	-0.154960	0.000257
-1.32	-0.166374	-0.165630	-0.165105	0.000214

Milstein results:

λ	min	mean	max	st dev
-1.80	0.064589	0.064960	0.065372	0.000145
-1.78	0.056201	0.056593	0.057021	0.000149
-1.76	0.047682	0.048068	0.048509	0.000152
-1.74	0.039087	0.039475	0.039963	0.000148
-1.72	0.030406	0.030794	0.031314	0.000151
-1.70	0.021576	0.022018	0.022556	0.000160
-1.68	0.012735	0.013140	0.013617	0.000165
-1.66	0.003844	0.004196	0.004656	0.000156
-1.64	-0.005221	-0.004846	-0.004449	0.000146
-1.62	-0.014366	-0.014013	-0.013650	0.000144
-1.60	-0.023695	-0.023277	-0.022902	0.000146
-1.58	-0.033071	-0.032645	-0.32237	0.000151
-1.56	-0.042511	-0.042125	-0.041703	0.000156
-1.54	-0.052140	-0.05172	-0.051286	0.000159
-1.52	-0.061865	-0.061436	-0.060998	0.000162
-1.50	-0.071721	-0.071282	-0.070874	0.000166
-1.48	-0.081665	-0.081257	-0.080847	0.000173
-1.46	-0.091848	-0.091377	-0.090851	0.000185
-1.44	-0.102095	-0.101645	-0.101088	0.000184
-1.42	-0.112209	-0.111845	-0.111292	0.000184
-1.40	-0.122637	-0.122206	-0.121697	0.000188
-1.38	-0.133379	-0.132915	-0.132461	0.000187
-1.36	-0.144237	-0.143822	-0.143381	0.000177
-1.34	-0.156505	-0.155577	-0.154960	0.000257
-1.32	-0.166131	-0.165607	-0.165042	0.000204

The corresponding linear regression formulae are follows:

For $\theta = 0$:

$$L_{mean} = -0.439580\lambda - 0.659155$$

$$L_0 = -1.499511$$

For $\theta = 0.5$:

$$L_{mean} = -0.458195\lambda - 0.723688$$

$$L_0 = -1.579432$$

For $\theta = 1$:

$$L_{mean} = -0.486329\lambda - 0.804402$$

$$L_0 = -1.654028$$

Even at a quick glance the results are not significantly different from the Euler-Maruyama ones for values of λ that are greater than the bifurcation value (the value where λ changes from negative to positive). However, the results deviate for the stable values of λ. Could this be a characteristic of the Lyapunov exponents rather than the numerical method? In other words, does using the Milstein method return more accurate values and it is the results of the Lyapunov exponents when the system turns chaotic that behave the same irrespective of the numerical method used? Or is it simply a characteristic of the system? That once the system passes a bifurcation point, the rate of divergence is the same?

To see if there are any further differences we investigate more of Norton's published equations and explorations and compare them with our Euler-Maruyama and Milstein ones:

Table of regression formulae published in [1]:

Equation	R
$L_{mean} = 0.4795h^2 + 0.00074$	0.954
$L_{mean} = 0.2641\sqrt{h} - 0.007847$	0.995
$L_{mean} = 0.2835h - 0.02506$	0.998
$L_{mean} = 0.170250h + 0.107215\sqrt{h} - 0.047085$	0.99993
$L_{mean} = -0.143577h^2 + 0.362442h - 0.031696$	0.999997
$L_{mean} = -0.1215886h^2 + 0.3323459h + 0.0170465\sqrt{h} - 0.0341816$	1
$L_{mean} = -0.2140995h^{1.5} + 0.4660743h - 0.0180839\sqrt{h} - 0.0309053$	1

Equations for our Euler-Maryama results:

Equation	R
$L_{mean} = 0.4760191h^2 + 0.0005359$	0.8952
$L_{mean} = 0.261823\sqrt{h} - 0.077958$	0.9875
$L_{mean} = 0.281224h - 0.025040$	0.9952
$L_{mean} = 0.175069h + 0.100497\sqrt{h} - 0.045681$	0.9998
$L_{mean} = -0.1351h^2 + 0.3555h - 0.03128$	1
$L_{mean} = -0.1259889h^2 + 0.3430317h + 0.0070656\sqrt{h} - 0.0323112$	1
$L_{mean} = -0.2212686h^{1.5} + 0.4807996h - 0.0289973\sqrt{h} - 0.0289602$	1

Equations for our Milstein results:

Equation	R
$L_{mean} = 0.4759483h^2 + 0.0005513$	0.9127
$L_{mean} = 0.261776\sqrt{h} - 0.077928$	0.9896
$L_{mean} = 0.281177h - 0.025020$	0.996
$L_{mean} = 0.175173h + 0.100354\sqrt{h} - 0.045631$	0.9998
$L_{mean} = -0.1349h^2 + 0.3554h - 0.03125$	1
$L_{mean} = -0.1258169h^2 + 0.3429066h + .0070497\sqrt{h} - 0.0322798$	1
$L_{mean} = -0.2209703h^{1.5} + 0.4804916h - 0.0289661\sqrt{h} - 0.0289331$	1

Although we see some difference between our results and Norton's, the differences between our Euler-Maruyama and Milstein results are negligable. So, the Milstein scheme exhibits the same relational behaviour between the step length the bifurcation approximation from the Lyapunov exponents. The following tables are simply more comparisons between our results from the Milstein scheme and Norton's original results, they all exhibit the same differences as seen before.

Table published in [2] ($\theta = 0$):

λ	Quadratic equation	R^2
-1.8	L=$-0.1109031h^2 + 0.303866h + 0.097956$	1
-1.7	L=$-0.126717h^2 + 0.322257h + 0.057182$	1
-1.6	L=$-0.133858h^2 + 0.341172h + 0.014109$	1
-1.5	L=$-0.143577h^2 + 0.362442h - 0.031696$	1
-1.4	L=$-0.152311h + 0.383909h - 0.080388$	1
-1.3	L=$-0.164308h^2$+$0.408497h - 0.132665$	1

Our Milstein results ($\theta = 0$):

λ	Quadratic equation	R^2
-1.8	L=$-0.1109h^2 + 0.2976h + 0.09836$	1
-1.7	L=$-0.12159h^2 + 0.31763h + 0.05745$	1
-1.6	L=$-0.1262h^2 + 0.3348h + 0.01452$	1
-1.5	L=$-0.1349h^2 + 0.3554h - 0.03125$	1
-1.4	L=$-0.07977h^2 + 0.3759h - 0.07977$	1
-1.3	L=$-0.15695h^2$+$0.40121h - 0.13198$	1

Norton's results:

λ	Bifurcation value of h
-1.8	No value
-1.7	No value
-1.6	No value
-1.5	0.0907
-1.4	0.2305
-1.3	0.3841

Our Milstein results:

λ	Bifurcation value of h
-1.8	No value
-1.7	No value
-1.6	No value
-1.5	0.0911
-1.4	0.2328
-1.3	0.3878

Following Norton's methodology further we then investigate the relationships between the Lyapunov exponents and both step length and λ. When plotted the results show an obvious relationship between the variables with the points forming a well structured plane as seen in Figure 5.

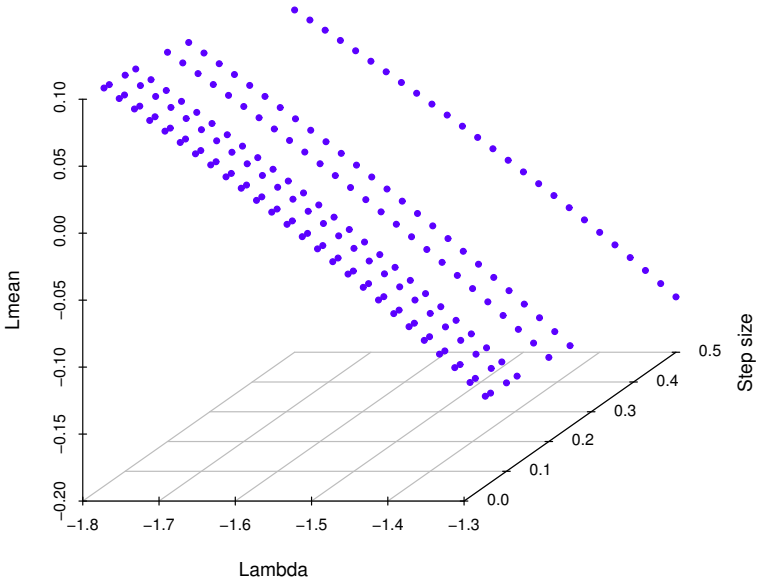

Figure 5: Milstein with $\theta = 0$

A simple linear regression yields the following equation:

$$L_{mean} = -0.42657\lambda + 0.27410h - 0.66626$$

When this plane is added to the plot it can be seen that there is a slight curvature that implies a quadratic fit would be better (Figure 6).

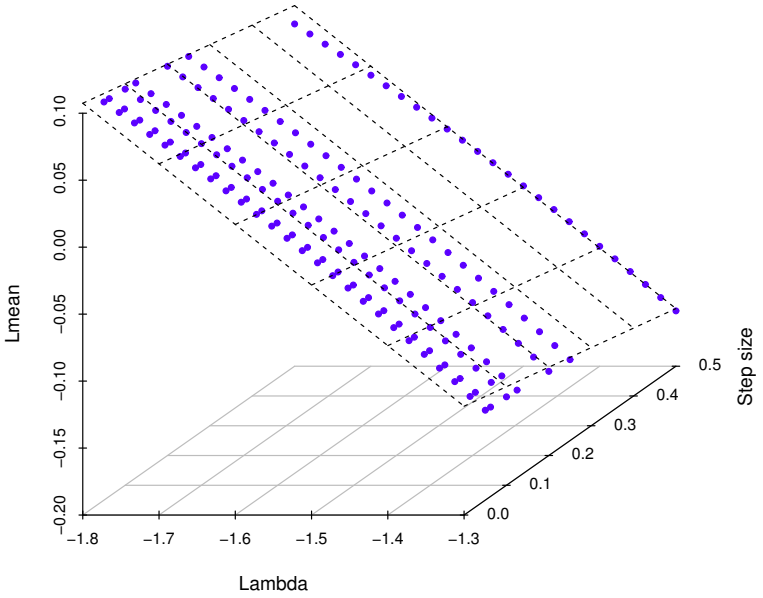

Figure 6: Milstein with $\theta = 0$ and regression plane

The quadratic fit indeed has an R value closer to 1. Norton presented quadratic formulas in both [1] and [2], the following tables show the comaprisons again between our results and his:

Quadratic regression formula published in [1] and [2] ($\mu = 0.1$):

θ	Equation	R
0	$L_{mean} = -0.13188\lambda - 0.13950h^2 - 0.83524\lambda - 0.35312h - 0.98643$	0.999
0.5	$L_{mean} = -0.13789\lambda - 0.13472h^2 - 0.88053\lambda - 0.00010h - 1.04244$	1

Our regression formula using Milstein ($\mu = 0.1$):

θ	Equation	R
0	$L_{mean} = -0.13015\lambda^2 - 0.1317h^2 - 0.83002\lambda - 0.34651h - 0.98209$	0.9979
0.5	$L_{mean} = -0.13474\lambda^2 - 0.12916h^2 - 0.87011\lambda - 0.00540h - 1.03355$	0.9994

We are starting to see some significant differences in our results, until now we had been within 2 decimal places of Norton's but for $\theta = 0.5$ we are not. This is where we would expect any differences to be most obvious due to the increased order of the Euler scheme when $\theta = 0.5$ and so the same with the Milstein scheme for $\theta = 0.5$.

15 Conclusion

One of the interesting observations from this investigation was the fact that the Milstein scheme retains the same degree of accuracy for all values of λ when compared to the results for the deterministic equation for, yet the Euler-Maruyama scheme is significantly less accurate for the stable regions (as expected) but they have similar values and deviations beyond the bifurcation point. The following table shows this for $h = 10$ (step length = 0.1):

	Milstein			Euler-Maruyama			Deterministic		
λ	$\theta = 0$	$\theta = 0.5$	$\theta = 1$	$\theta = 0$	$\theta = 0.5$	$\theta = 1$	$\theta = 0$	$\theta = 0.5$	$\theta = 1$
-1.30	-0.09343	-0.13397	-0.17700	-0.09418	-0.14306	-0.17784	-0.09350	-0.13403	-0.17705
-1.32	-0.08313	-0.12318	-0.16561	-0.08545	-0.12410	-0.16563	-0.08322	-0.12325	-0.16562
-1.34	-0.07301	-0.11258	-0.15558	-0.08168	-0.11273	-0.15558	-0.07310	-0.11264	-0.15498
-1.36	-0.06305	-0.10224	-0.14382	-0.06635	-0.10227	-0.14766	-0.06313	-0.10226	-0.14388
-1.38	-0.05323	-0.09218	-0.13292	-0.05554	-0.09235	-0.13349	-0.05332	-0.09233	-0.13297
-1.40	-0.04356	-0.08213	-0.12221	-0.04558	-0.08282	-0.12227	-0.04365	-0.08221	-0.12225
-1.42	-0.3402	-0.07214	-0.11184	-0.03613	-0.07363	-0.11190	-0.03412	-0.07222	-0.11189
-1.44	-0.02460	-0.06229	-0.10165	-0.02721	-0.06470	-0.10216	-0.02472	-0.06238	-0.10172
-1.46	-0.01530	-0.05258	-0.09138	-0.01934	-0.05586	-0.09307	-0.01544	-0.05268	-0.09145
-1.48	-0.00612	-0.04300	-0.08126	-0.01184	-0.04669	-0.08531	-0.00627	-0.04310	-0.08133
-1.50	0.00296	-0.03355	-0.07128	0.00294	-0.03691	-0.07621	0.00279	-0.03366	-0.07135
-1.52	0.01192	-0.02421	-0.06144	0.01191	-0.02681	-0.06440	0.01177	-0.02434	-0.06151
-1.54	0.02078	-0.01499	-0.05172	0.02077	-0.01679	-0.05414	0.02064	-0.01513	-0.05180
-1.56	0.02955	-0.00587	-0.04213	0.02953	-0.00701	-0.04449	0.02942	-0.00603	-0.04222
-1.58	0.03821	0.00314	-0.03265	0.03820	0.00313	-0.03536	0.03810	0.00297	-0.03275
-1.60	0.04678	0.01206	-0.02328	0.04677	0.01205	-0.02705	0.04669	0.01190	-0.02341
-1.62	0.05516	0.02088	-0.01401	0.05515	0.02086	-0.02322	0.05508	0.02073	-0.01417
-1.64	0.06346	0.02959	-0.00485	0.06345	0.02957	-0.00799	0.06336	0.02946	-0.00503
-1.66	0.07175	0.03820	0.00420	0.07174	0.03819	0.00418	0.07167	0.03811	0.00402
-1.68	0.07997	0.04674	0.01314	0.07996	0.04673	0.01313	0.07990	0.04667	0.01299
-1.70	0.08811	0.05505	0.02202	0.08810	0.05503	0.02200	0.08806	0.05493	0.02187
-1.72	0.09585	0.06339	0.03079	0.09584	0.06338	0.03078	0.09576	0.06329	0.03066
-1.74	0.10382	0.07167	0.03947	0.10381	0.07166	0.03946	0.10376	0.07158	0.03936
-1.76	0.11173	0.07986	0.04807	0.11173	0.07985	0.04806	0.11168	0.07980	0.04798
-1.78	0.11951	0.08798	0.05659	0.11950	0.08797	0.05658	0.11950	0.08795	0.05652
-1.80	0.12692	0.09575	0.06496	0.12691	0.09573	0.06495	0.12684	0.09795	0.06490

The Milstein scheme has the same relationship between the approximations and step length as the Euler-Maruyama scheme. So the two methods behave the same when applied to this method but Milstein has greater accuracy up to and including the bifuraction point.

If we are only interested in calculating moments and statistics of a system then there is no advantage to using a Milstein scheme over an Euler-Maruyama (as stated in the section 9), if we are interested in a pathwise fit then Milstein maintains its higher accuracy up to and including the bifurcation point but both schemes are the same once the systems passes the bifurcation point.

16 Further research and discussion

On first reading of Ford and Norton's paper [1] we initially wondered if the distribution of the Lyapunov exponent would be distributed in line with the type of noise experienced in the system, would a gamma distributed noise results in gamma distributed Lyapunov's? This was answered during the investigations for this paper. The noise in the SDDE used here is not enough to skew the entire system. If a system was governed by the distribution of the stochastic elements then the Lyapunov exponents are likely to fail due to the fact that they detect deterministic chaos and return a guaranteed positive in purely random systems. Also, as the methodology is constructed from running multiple simulations then the Lyapunov exponents are simply results from repeated experiments. Even if the system was naturally skewed the Lyapunov exponents would still be normally distributed round the expected value for the system at those parameters.

There are many ways of furthering the work conducted in this paper, the first of which would be examining different forms of SDDE's or other numerical methods. We used a linear SDDE with a fixed delay and an additive Gaussian/Brownian stochastic term. We could look at the effects of variable delays, mixed delays or where the stochastic term is part of the delay to see if the same relational behaviours exist. We could examine other numerial methods, again to see if the finding of this paper hold for other methods or if there are some that deviate in any significant way. We could also look at the influence of different types of noise in greater detail, this could entail applying different methods not covered in this paper such as Malliavin calculus (instead of Ito) and different methods for assessing the bifurcation points that are less prone to erroneous solutions than the Lyapunov exponents (false positives for chaos in the presence of too much randomness). We suspect that simply increasing the level of noise would increase the deviations in the approximation but other types of noise may yield something interesting. Works such as Lu and Ding [26] have shown that 'jumps' in a model can affect the system significantly and Zhao and Lui [32] investigated an SDDE with Poisson jumps using a backwards Euler method. Bocharov and Rihan [27] presented an HIV model

with Gamma distributed delays. We could also look at nonlinear systems such as Santonia and Shalkhet [22], who examined an obesity epidemic via a nonlinear stochastic delay model with additive noise and an Euler-Maruyama scheme.

One of the particularly interesting prospects of this work is the ability to apply the techniques in applied areas. There appear to be many applications that have used Euler-Maruyama schemes, some of which may benefit from the increased accuracy of using a Mistein scheme without any overall difference to the behaviour or computation. Carletti's paper [21] in particular uses a linear SDDE with stochastic perturbations and Lyapunov exponents to examine the coexistence of phage and bacteria in work examining marine bacteriophage. Carletti found that the longer the incubation time, the longer the two can coexist in equilibrium in a noisy enviroment. Several papers on this topic [21,29,39] claim that this area of study is of particular interest due to the prospect of using phage as an alternative to antibiotic therapy in bacterial infections and Lui, Lui and Tang [39], claims that one question that remains outstanding is under which conditions they will coexist permanently.

Another topical area of interest is ecological and enviromental. The Azimuth project [34] includes SDDE's as part of its ongoing working studying the El Nino Southern Oscillator phenomenon where SDDE's can build on the work of Richard Kleeman's "Stochastic theories for the irregularity of ENSO" [28]. There also seems to be potential in areas such as Lui, Gang, Yang, Junhui, Jigang and Lei's paper on predicting wind power [30] where SDDE's were not used but could be.

Medicine and biology is another field where the popularity of using SDDE's as a modelling tool is emerging. Population dynamics continually appear in recent research papers such as de la Hoz and Vadillo [33] and Xia, Jiang and Li [37] and the techniques from these translate to studies in epidemics and diseases. Examples of which include Li, Sun and Jin [24], Krsti´c [36] and Saker [25] . In medicine there also seems to be a preference for using Gillespie algorithms, a variation of Monte Carlo simulations designed for simulating chemical or biological systems, perhaps there is merit in understanding these methods and whether they could be combined with some of the concepts included here. Ion and Georgescu [31] studied bifurcations in a leukemia model using Lyapunov exponents, this combined with Brent Neiman's slightly older work [23] using delays to account for unobservable or slow moving changes in a three stage mathematical model of chronic myelogenous leukemia is yet another medical application that could potentially benefit significantly with further research using some of the techniques we have used, in particular by adding stochastic elements to account for triggering the transition of the model to the chronic stage.

So far we have not cited examples from economics and finance, engineering or physics. These

fields are prolific in examples of DDE's, SDE's and SDDE's, however many of the examples mentioned here have been published within the last year and go some way to illustrate the current efforts using the techniques presented in this paper. Advances in technology is especially allowing greater experimentation with stochastics and delay equations that is building a far greater understanding of concepts that were relatively undocumented in previous decades.

17 Appendix

Code for generating the Gaussian numbers:

```
Random random = new java.security.SecureRandom()
i=0
while (i<500){
i++
f = new File(i + 'Gaussian' + '.txt')
n = 100000
for (i in 1..n){
BM = random.nextGaussian()
f.append(BM + '\n')
}
}
```

R scripts:

```
Hdata<-read.table("hdata.txt", sep="\t", header=F, col.names=c("steps","LE"))
model<-lm(Hdata$T0~I(Hdata$steps^2))
model<-lm(Hdata$T0~I(sqrt(Hdata$steps)))
model<-lm(Hdata$T0~Hdata$steps)
model<-lm(Hdata$T0~Hdata$steps + I(sqrt(Hdata$steps)))
model<-lm(Hdata$T0~Hdata$steps + I(Hdata$steps^2))
model<-lm(Hdata$T0~Hdata$steps + I(Hdata$steps^2) + I(sqrt(Hdata$steps)))
model<-lm(Hdata$T0~Hdata$steps + I(Hdata$steps^1.5) + I(sqrt(Hdata$steps)))
summary(model)
```

```
library(scatterplot3d)
mydata<-read.table("ScatterData05.txt",header=F,sep="\t",col.names=c("x","y","z"))
fit<-lm(mydata$z~mydata$x + mydata$y)
s3d<-scatterplot3d(mydata$x,mydata$y,mydata$z,type="p",color="blue",pch=20, main ="Scatterplot",
xlab="Lambda",ylab="Step size",zlab="Lmean",box=F, angle=-50)
s3d$plane3d(fit)
```

Full Milstein results for $\theta = 0$

λ	0.5	0.25	0.2	0.125	0.1	0.0625	0.05
-1.30	0.02939	-0.04159	-0.05798	-0.08413	-0.09343	-0.10784	-0.11212
-1.32	0.03825	-0.03200	-0.04816	-0.07391	-0.08313	-0.09692	-0.10202
-1.34	0.04700	-0.02250	-0.03849	-0.06410	-0.07301	-0.08682	-0.09160
-1.36	0.05561	-0.01316	-0.02896	-0.05419	-0.06305	-0.07692	-0.08140
-1.38	0.06410	-0.00392	-0.01956	-0.04444	-0.05323	-0.06665	-0.07159
-1.40	0.07249	0.00519	-0.01028	-0.03492	-0.04356	-0.05686	-0.06139
-1.42	0.08077	0.01419	-0.00112	-0.02551	-0.3402	-0.04735	-0.05187
-1.44	0.08895	0.02308	0.00795	-0.01617	-0.02460	-0.03749	-0.04196
-1.46	0.09693	0.03185	0.01689	-0.00695	-0.01530	-0.02812	-0.03265
-1.48	0.10492	0.04053	0.02573	0.00214	-0.00612	-0.01890	-0.02307
-1.50	0.11270	0.04912	0.03442	0.01111	0.00296	-0.00951	-0.01391
-1.52	0.12047	0.5758	0.04305	0.01999	0.01192	-0.00058	-0.00470
-1.54	0.12811	0.06592	0.05160	0.02878	0.02078	0.00861	0.00443
-1.56	0.13562	0.07415	0.05994	0.03744	0.02955	0.01746	0.01324
-1.58	0.14303	0.08224	0.06830	0.04599	0.03821	0.02624	0.02234
-1.60	0.15037	0.09028	0.07642	0.05445	0.04678	0.03502	0.03082
-1.62	0.15763	0.09826	0.08460	0.06281	0.05516	0.04345	0.03973
-1.64	0.16480	0.10615	0.09254	0.07108	0.06346	0.05211	0.04804
-1.66	0.17190	0.11396	0.10054	0.07926	0.07175	0.06029	0.05645
-1.68	0.17893	0.12152	0.10827	0.08736	0.07997	0.06881	0.06498
-1.70	0.18589	0.12914	0.11615	0.09536	0.08811	0.07677	0.07295
-1.72	0.19278	0.13672	0.12370	0.10323	0.09585	0.08516	0.08151
-1.74	0.19958	0.14406	0.13132	0.11094	0.10382	0.09292	0.08931
-1.76	0.20617	0.15141	0.13884	0.11856	0.11173	0.10114	0.09705
-1.78	0.21288	0.15877	0.14615	0.12623	0.11951	0.10876	0.10543
-1.80	0.21944	0.16581	0.15350	0.13385	0.12692	0.11665	0.11291

Full Milstein results for $\theta = 0.5$

λ	0.5	0.25	0.2	0.125	0.1	0.0625	0.05
-1.30	-0.15868	-0.13953	-0.13718	-0.13450	-0.13397	-0.13329	-0.13285
-1.32	-0.14874	-0.12903	-0.12651	-0.12392	-0.12318	-0.12230	-0.12258
-1.34	-0.13895	-0.11869	-0.11625	-0.11336	-0.11258	-0.11196	-0.11150
-1.36	-0.12932	-0.10851	-0.10589	-0.10296	-0.10224	-0.10179	-0.10143
-1.38	-0.11984	-0.09847	-0.09572	-0.09285	-0.09218	-0.09120	-0.09114
-1.40	-0.11049	-0.08858	-0.08581	-0.08267	-0.08213	-0.08110	-0.08093
-1.42	-0.10117	-0.07874	-0.07598	-0.07290	-0.07214	-0.07132	-0.07118
-1.44	-0.09207	-0.06907	-0.06622	-0.06303	-0.06229	-0.06131	-0.06101
-1.46	-0.08303	-0.05957	-0.05661	-0.05329	-0.05258	-0.05156	-0.05154
-1.48	-0.07417	-0.05011	-0.04713	-0.04378	-0.04300	-0.04215	-0.04165
-1.50	-0.06537	-0.04084	-0.03777	-0.03439	-0.03355	-0.03244	-0.03240
-1.52	-0.05667	-0.03164	-0.02852	-0.02504	-0.02421	-0.02314	-0.02282
-1.54	-0.04809	-0.02258	-0.01939	-0.01582	-0.01499	-0.01403	-0.01371
-1.56	-0.03961	-0.01361	-0.01037	-0.00671	-0.00587	-0.00473	-0.00449
-1.58	-0.03123	-0.00474	-0.00144	0.00227	0.00314	0.00413	0.00459
-1.60	-0.02293	0.00404	0.00740	0.01114	0.01206	0.01323	0.01338
-1.62	-0.01474	0.01272	0.01614	0.01994	0.02088	0.02204	0.02246
-1.64	-0.00664	0.02130	0.02479	0.02863	0.02959	0.03072	0.03094
-1.66	0.00139	0.02983	0.03335	0.03721	0.03820	0.03945	0.03985
-1.68	0.00936	0.03824	0.04176	0.04571	0.04674	0.04782	0.04814
-1.70	0.01726	0.04656	0.05014	0.05411	0.05505	0.05642	0.05662
-1.72	0.02507	0.05478	0.05845	0.06242	0.06339	0.06455	0.06506
-1.74	0.03279	0.06291	0.06656	0.07064	0.07167	0.07303	0.07304
-1.76	0.04044	0.07097	0.07471	0.07877	0.07986	0.08095	0.08162
-1.78	0.04802	0.07894	0.08268	0.08683	0.08798	0.08929	0.08942
-1.80	0.05553	0.08683	0.09063	0.09481	0.09575	0.09702	0.09727

References

[1] Ford NJ, Norton SJ. Noise-induced changes to the behaviour of semi-implicit Euler methods for stochastic delay differential equations undergoing bifurcation. J Comput Appl Math 2009 JUL 15 2009;229(2):462-470

[2] Norton, S. J. Noise induced changes to dynamic behaviour of stochastic delay differential equations. (Unpublished doctoral dissertation). University of Liverpool, United Kingdom; 2008

[3] Wulf, V. Numerical Analysis of Delay Differential Equations Undergoing a Hopf Bifurcation. University of Liverpool, United Kingdom; 1999

[4] Raymond Brummelhuis. Department of Economics, Mathematics and statistics, Birkbeck University. Mathematical Methods Lecture Notes. http://www.ems.bbk.ac.uk/for_students/msc_finEng/math_methods/ ; 2014

[5] Higham DJ. An algorithmic introduction to numerical simulation of stochastic differential equations. SIAM Rev 2001 SEP 2001;43(3):525-546

[6] Arnold L. Random Dynamical Systems. : Springer Berlin Heidelberg; 2010

[7] Kloeden PE, Platen E. Numerical Solution of Stochastic Differential Equations. : Springer; 1992

[8] Hale JK, Buttanri H, Kocak H. Dynamics and Bifurcations. : Springer New York; 2011

[9] Wolf A, Swift JB, Swinney HL, Vastano JA. Determining Lyapunov exponents from a time series. Physica D 1985;16(3):285-317.

[10] Wolff R, Yao Q, Tong H. Statistical tests for Lyapunov exponents of deterministic systems. Studies in nonlinear dynamics & econometrics 2004;8(2).

[11] Milshtein.GN. Approximate Integration of Stochastic Differential Equations. Teoriya Veroyatnostei i Yeye Primeniya 1974 1974;19(3):583-588

[12] Dingwell JB. Lyapunov Exponents. Wiley Encyclopedia of Biomedical Engineering: John Wiley & Sons, Inc.; 2006

[13] Liu HF, Dai ZH, Li WF, Gong X, Yu ZH. Noise robust estimates of the largest Lyapunov exponent. Physics Letters a 2005 JUN 20 2005;341(1-4):119-127

[14] Yonemoto K, Yanagawa T. Estimating the Lyapunov exponent from chaotic time series with dynamic noise. Statistical Methodology 2007 10;4(4):461-480

[15] Argyris J, Andreadis I. On the influence of noise on the largest Lyapunov exponent of attractors of stochastic dynamic systems. Chaos Solitons & Fractals 1998 JUN 1998;9(6):959-963.

[16] |Serletis A, Shahmoradi A, Serletis D. Effect of noise on estimation of Lyapunov exponents from a time series. Chaos Solitons & Fractals 2007 APR 2007;32(2):883-887

[17] Dennis B, Desharnais RA, Cushing JM, Henson SM, Costantino RF. Can noise induce chaos? Oikos 2003 AUG 2003;102(2):329-339

[18] Kostelich EJ, Yorke JA. Noise-Reduction - Finding the Simplest Dynamic System Consistent with the Data. Physica D 1990 MAR 1990;41(2):183-196

[19] Franca LFP, Savi MA. Evaluating noise sensitivity on the time series determination of lyapunov exponents applied to the nonlinear pendulum. Shock Vibrat 2003 2003;10(1):37-50

[20] Kloeden PE, Platen E, Schurz H. Numerical Solution of SDE Through Computer Experiments. : Springer Berlin Heidelberg; 1994.

[21] Carletti M. Mean-square stability of a stochastic model for bacteriophage infection with time delays. Math Biosci 2007;210(2):395-414.

[22] Santonja F, Shaikhet L. Probabilistic stability analysis of social obesity epidemic by a delayed stochastic model. Nonlinear Analysis: Real World Applications 2014;17:114-125.

[23] Neiman B. A Mathematical Model of Chronic Myelogenous Leukemia. Oxford: University College Oxford University; 2000

[24] Li L, Sun G, Jin Z. Bifurcation and chaos in an epidemic model with nonlinear incidence rates. Applied Mathematics and Computation 2010 APR 15;216(4):1226-1234.

[25] Saker S. Stability and Hopf bifurcations of nonlinear delay malaria epidemic model. Nonlinear Analysis: Real World Applications 2010;11(2):784-799.

[26] Lu C, Ding X. Permanence and Extinction of a Stochastic Delay Logistic Model with Jumps. Mathematical Problems in Engineering 2014;2014.

[27] Bocharov GA, Rihan FA. Numerical modelling in biosciences using delay differential equations. J Comput Appl Math 2000 DEC 15 2000;125(1-2):183-199.

[28] Kleeman R. Stochastic theories for the irregularity of ENSO. Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences 2008 JUL 28 2008;366(1875):2511-2526.

[29] E. Beretta, Y. Kuang, Modeling and analysis of a marine bacteriophage infection, Math. Biosci. 149 (1998) 57–76.

[30] Yan Gangui, Liu Yu, Mu Gang, Cui Yang, Li Junhui, Liu Jigang, et al. The Ultra-short Term Prediction of Wind Power Based on Chaotic Time Series. 2012 International Conference on Future Electrical Power and Energy System, Pt B 2012 2012;17:1490-1496.

[31] Ion AV, Georgescu RM. Bautin bifurcation in a delay differential equation modeling leukemia. Nonlinear Analysis-Theory Methods & Applications 2013 APR 2013;82:142-157.

[32] Zhao G, Liu M. Numerical methods for nonlinear stochastic delay differential equations with jumps. Applied Mathematics and Computation 2014;233:222-231.

[33] de la Hoz F, Vadillo F. A mean extinction-time estimate for a stochastic Lotka–Volterra predator–prey model. Applied Mathematics and Computation 2012;219(1):170-179.

[34] The Azimuth Project. Stochastic delay differential equations: http://www.azimuthproject.org/azimuth/show/Stochastic+delay+differential+equations ; 2014

[35] Li L, Sun G, Jin Z. Bifurcation and chaos in an epidemic model with nonlinear incidence rates. Applied Mathematics and Computation 2010 APR 15;216(4):1226-1234.

[36] Krstić M. The effect of stochastic perturbation on a nonlinear delay malaria epidemic model. Math Comput Simul 2011;82(4):558-569

[37] Xia P, Jiang D, Li X. Dynamical Behavior of the Stochastic Delay Mutualism System. Abstract and Applied Analysis. 2014; 2014: 19 pages.

[38] Prof. dr Andrzej Palczewski. Section of Equations of Mathematical Physics, University of Warsaw. Numerical schemes for SDEs Lecture Notes 5: http://www.mimuw.edu.pl/~apalczew/CFP_lecture5.pdf ; 2014

[39] Liu S, Liu Z, Tang J. A delayed marine bacteriophage infection model. Applied Mathematics Letters 2007;20(6):702-706.