Intro to UQ Coursework 2023/24

Dylan Besson

Student ID: 10724837

1 Question 1a

The forward Euler approximations of (1) are:

$$X_{i+1} = X_i + \Delta t \left(\alpha_1 - \beta_1 X_i \frac{Z_i^n}{K^n + Z_i^n} \right)$$

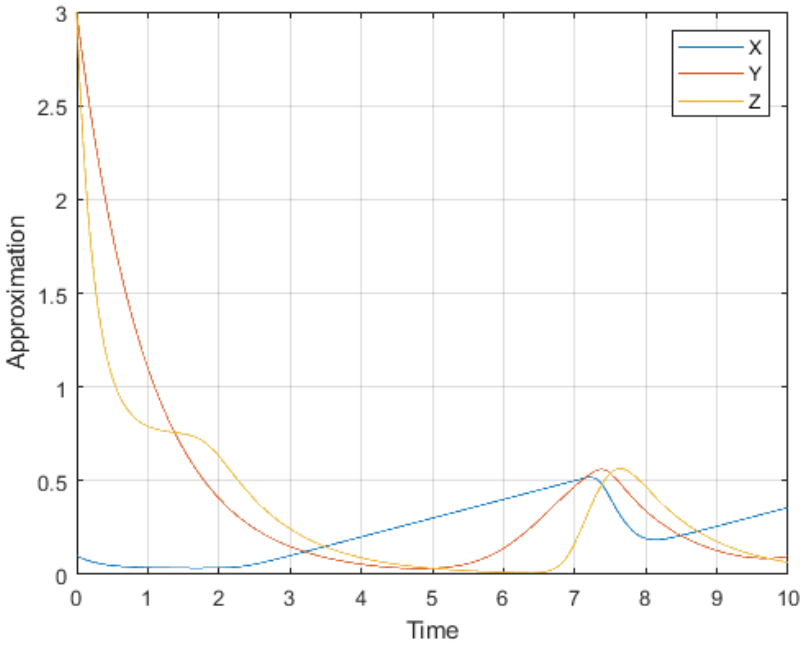
$$Y_{i+1} = Y_i + \Delta t \left(\alpha_2 (1 - Y_i) \frac{X_i^n}{K^n + X_i^n} - \beta_2 Y_i \right)$$

$$Z_{i+1} = Z_i + \Delta t \left(\alpha_3 (1 - Z_i) \frac{Y_i^n}{K^n + Y_i^n} - \beta_3 Z_i \right)$$

where $t_i = i\Delta t$, $X_i \approx X(i\Delta t)$, $Y_i \approx Y(i\Delta t)$ and $Z_i \approx Z(i\Delta t)$.

The forwardEuler.m function (code shown in appendix section 5.1) takes all the parameters (α_1, α_2, etc.) and implements the forward Euler approximations of (1). It returns each $X_0, X_1, ..., Y_0, Y_1, ..., Z_0, Z_1, ...$ in the vectors X, Y and Z, until the final time T is reached.

The code in appendix section 5.2 plots the approximation with the values of the parameters given in the brief, with timestep length $\Delta t = 10^{-5}$. The approximations for $x(10), y(10)$ and $z(10)$ are 0.3572, 0.0975 and 0.0648 respectively.

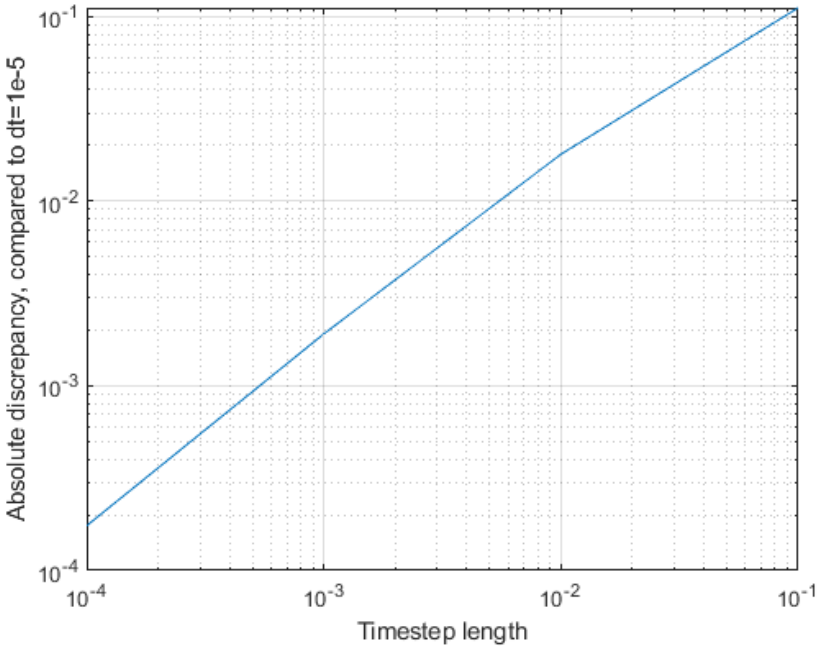


Then, for each of the timestep lengths mentioned in the brief, we calculate the absolute discrepancy in their final approximations for $x(10), y(10)$ and $z(10)$, compared to that of the original timestep length $\Delta t = 10^{-5}$:

$$\sqrt{(X(10) - X_a(10))^2 + (Y(10) - Y_a(10))^2 + (Z(10) - Z_a(10))^2}$$

where $X(10), Y(10)$ and $Z(10)$ are the final approximations with timestep length $\Delta t = 10^{-5}$, and $X_a(10), Y_a(10)$ and $Z_a(10)$ are the final approximations with timestep length $\Delta t = 10^{-a}$.

Finally, we plot a log-log graph, with timestep lengths on the x-axis, and their absolute discrepancies on the y-axis. It is natural to use a base-10 logarithmic scale for the timestep axis, because the timesteps are increasing by powers of 10. A log scale is used for the absolute discrepancy axis for better readability.



2 Question 1b

Please note that my interpretation of the question is to use timestep length $\Delta t = 10^{-2}$ throughout, but the code could easily be adjusted to use other timesteps if that was required.

I will denote the Monte-Carlo estimator for $\mathbb{E}(x(10))$ by

$$X_N = \frac{1}{N}\sum_{j=1}^{N} X_j(10),$$

where $X_1(10), X_2(10), ..., X_N(10)$ are the approximated values of $X(10)$, obtained using the forward Euler approximation algorithm designed in Q1a.

MC estimators Y_N and Z_N are defined in a similar way.

2.1 The Monte-Carlo Algorithm

The monteCarloQ1.m function (appendix section 5.3) takes parameter N, the number of samples it should use to calculate the MC estimates. It randomly picks the initial conditions as specified, throwing away any negative ones, then uses the forward Euler approximation algorithm from part Q1a. It returns 3 vectors X, Y and Z, whose entries are the N observations for $x(10), y(10)$ and

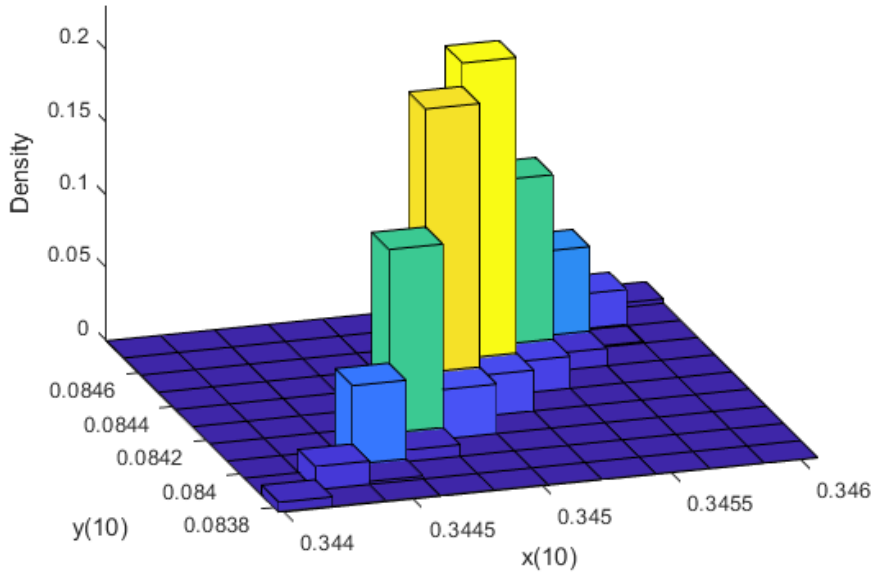
$z(10)$. It also returns the MC estimates X_N, Y_N and Z_N, calculated according to the formula above.

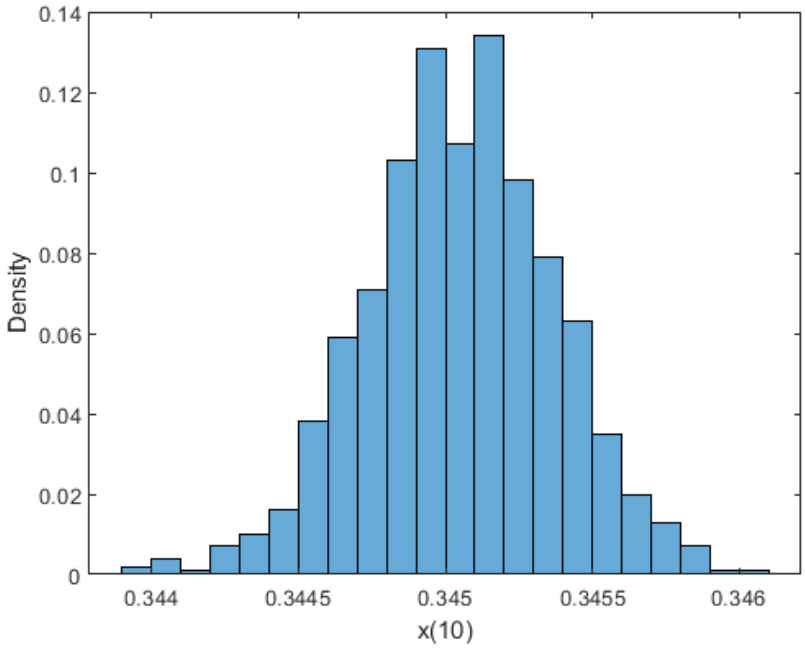
2.2 Distribution Plots and MC Estimates

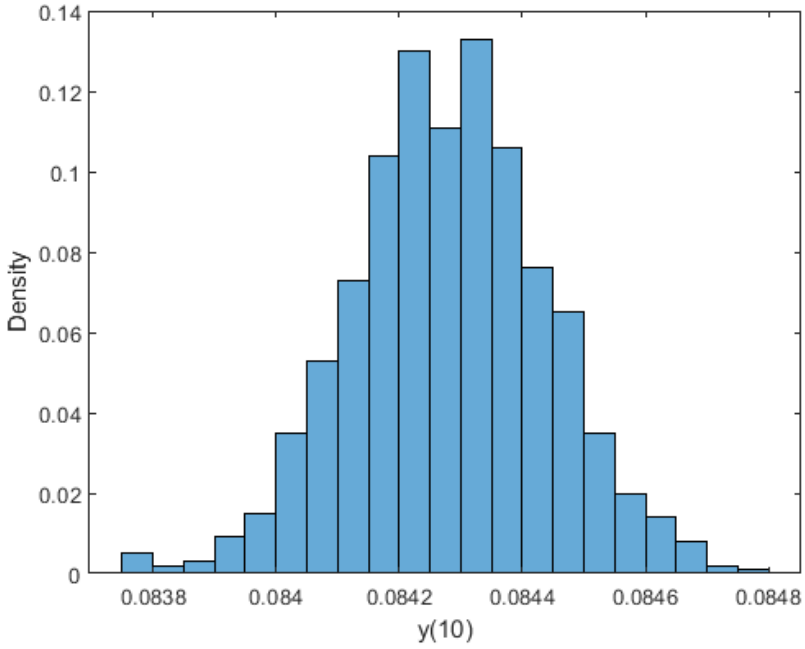
The code in appendix section 5.4 sets outputs our MC estimates X_N, Y_N and Z_N, then plots the joint and conditional distributions of $x(10)$ and $y(10)$.

For the MC estimates, with $N = 1000$ samples and timestep length $\Delta t = 10^{-2}$, we get 0.3451, 0.0843 and 0.0653 respectively. Note that these estimates vary very slightly due to the inherent randomness in the initial conditions, but not by much.

Here are the joint and conditional distribution plots:




0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Density
0.344
0.3445
0.345
0.3455
0.346
x(10)


0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Density
0.0838
0.084
0.0842
0.0844
0.0846
0.0848
y(10)

2.3 Standard Error

We work with X only for now, to explain how to tackle this part of the question. The standard error of X_N is given by its standard deviation. We want this to be less than 10^{-3}:

$$s.e.(X_N) = \sqrt{Var(X_N)} \leq 10^{-3}$$

We also know that

$$Var(X_N) = \frac{1}{N} Var(\hat{X}(10)),$$

where $\hat{X}(10)$ is our numerical approximation for x(10). This result is given by a proposition in the notes.

We cannot directly calculate $Var(\hat{X}(10))$, but we can approximate it by taking some n observations from $\hat{X}(10)$, and calculating their sample variance:

$$\hat{V}_n = \frac{1}{n-1} \left(\sum_{j=1}^{n} X_j(10)^2 - \left(\frac{1}{n} \sum_{j=1}^{n} X_j(10)^2 \right) \right)$$

I chose to calculate this sample variance with n = 1000, since we have already saved these observations from earlier in the question. It saves having to repeat all the same calculations! With n = 1000, we get $\hat{V}_n \approx 0.1191$. The code in appendix section 5.5 shows how the sample variance is calculated. With this, we can approximate our standard error, then find a lower bound on N:

$$s.e.(X_N) \approx \sqrt{\frac{1}{N} \hat{V}_n} \leq 10^{-3}$$

$$\Rightarrow N \geq 10^6 \hat{V}_n \approx 119,100$$

after some simple rearranging. This gives us an approximate lower bound on N, the number of samples, in order to keep the standard error below 10^{-3}.

We can repeat the exact same argument for Y and Z. On doing so, we get the lower bounds as approximately 7100 and 4300 respectively.

Note that these approximations on the lower bounds for N are rough estimates, mostly due to the randomness present throughout the calculations.

3 Question 2a

The Euler-Maruyama approximations are given by

$$X_{i+1} = X_i + \Delta t \left(\alpha_1 - \beta_1 X_i \frac{Z_i^n}{K^n + Z_i^n} \right) + \gamma X_i \sqrt{\Delta t} \omega_i$$

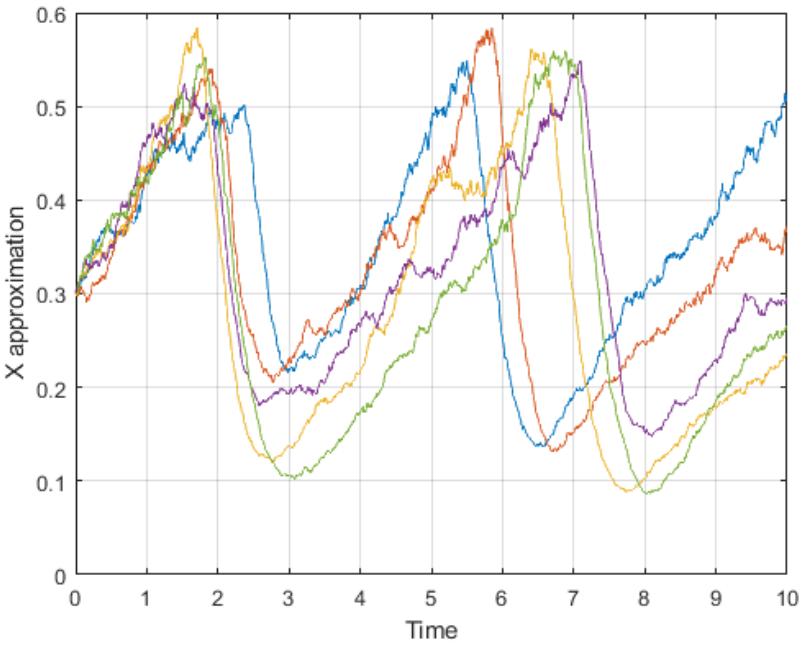
$$Y_{i+1} = Y_i + \Delta t \left(\alpha_2 (1 - Y_i) \frac{X_i^n}{K^n + X_i^n} - \beta_2 Y_i \right) + \gamma Y_i \sqrt{\Delta t} \omega_i$$

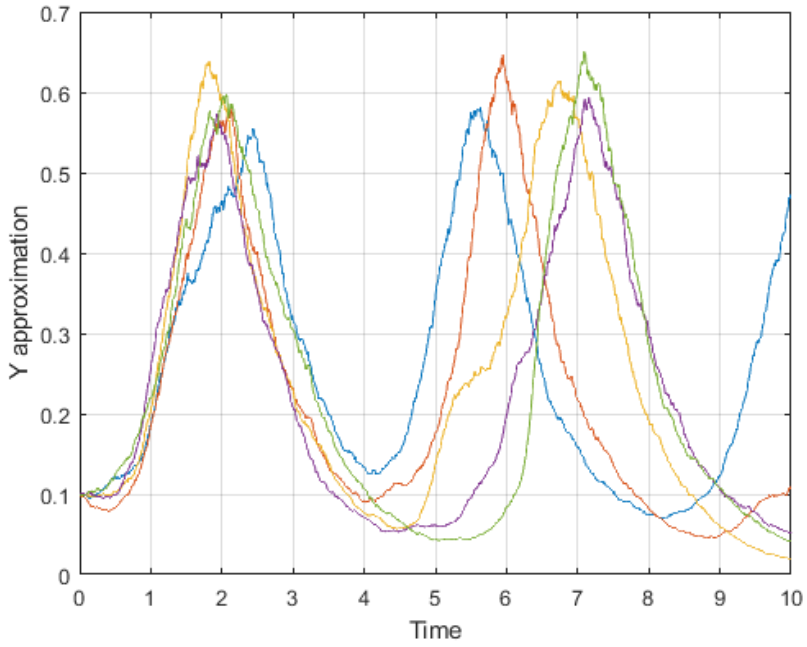
$$Z_{i+1} = Z_i + \Delta t \left(\alpha_3 (1 - Z_i) \frac{Y_i^n}{K^n + Y_i^n} - \beta_3 Z_i \right) + \gamma Z_i \sqrt{\Delta t} \omega_i$$

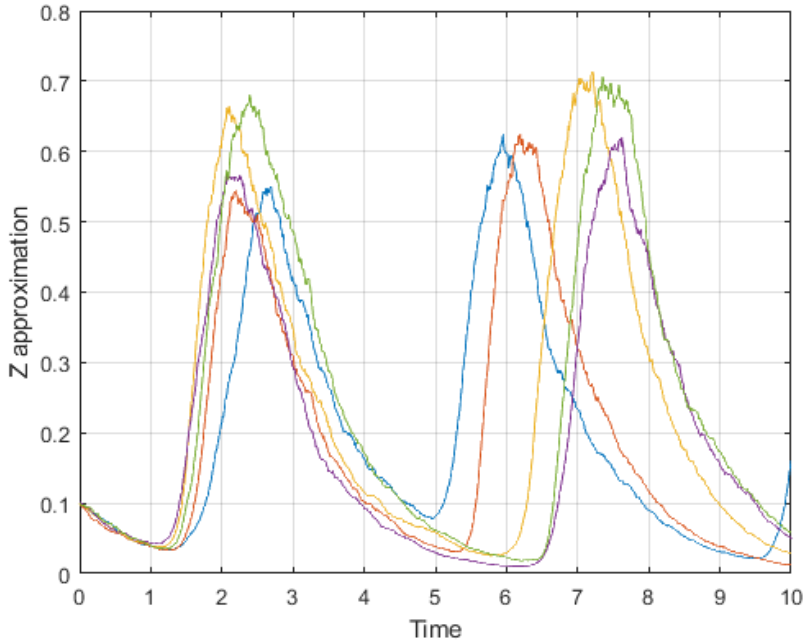
where $t_i = i\Delta t$, $X_i \approx X(i\Delta t)$, $Y_i \approx Y(i\Delta t)$, $Z_i \approx Z(i\Delta t)$ and $\omega_i \sim N(0,1)$ are i.i.d. random variables.

The eulerMaruyama.m function (given in appendix section 5.6) is very similar to the forwardEuler.m function from earlier. The differences are the one extra parameter in γ, the need to sample $\omega_i \sim N(0,1)$ for each iteration, and the use of the Euler-Maruyama approximations above, instead of the forward Euler ones.

The code in appendix section 5.7 applies and plots the Euler-Maruyama approximations 5 times, with the given initial condition values, $T = 10$ and $\Delta t = 10^{-2}$. I decided to plot the X, Y and Z approximations separately for clarity.







In Q1a, the ODE trajectories will always be the same, since there are no random terms. In Q1b, the ODE trajectories vary due to the randomness in the initial

conditions. Since the randomness is only in the initial conditions, the trajectories will look very similar throughout. i.e. they follow the same patterns. They only differ slightly in their values.

In this question, there is inherent randomness in the ω term of the Euler-Maruyama approximations. This randomness applies at every single timestep, not just at the start. So, the trajectories still largely follow the same pattern, but they vary more throughout. This can be seen when we plot the same approximations 5 times over. The 5 trajectories somewhat agree for about 2 seconds, then split off. They all follow a similar oscillating pattern, but their periods vary and their paths are jagged.

4 Question 2b

In a similar way to Q1b, denote the Monte-Carlo estimate for $\mathbb{E}(x(10))$ by

$$X_N = \frac{1}{N}\sum_{j=1}^{N} X_j(10)$$

where $X_1(10), X_2(10), ..., X_N(10)$ are the approximated values of $X(10)$, obtained using the Euler-Maruyama approximation algorithm designed in Q2a.

MC estimators Y_N and Z_N for $\mathbb{E}(y(10))$ and $\mathbb{E}(z(10))$ are defined in a similar way.

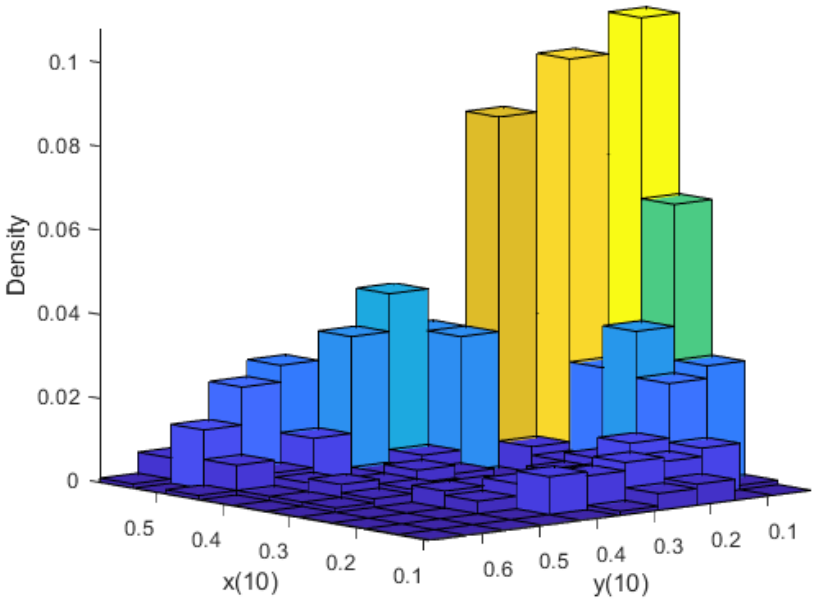
4.1 The Monte-Carlo Algorithm

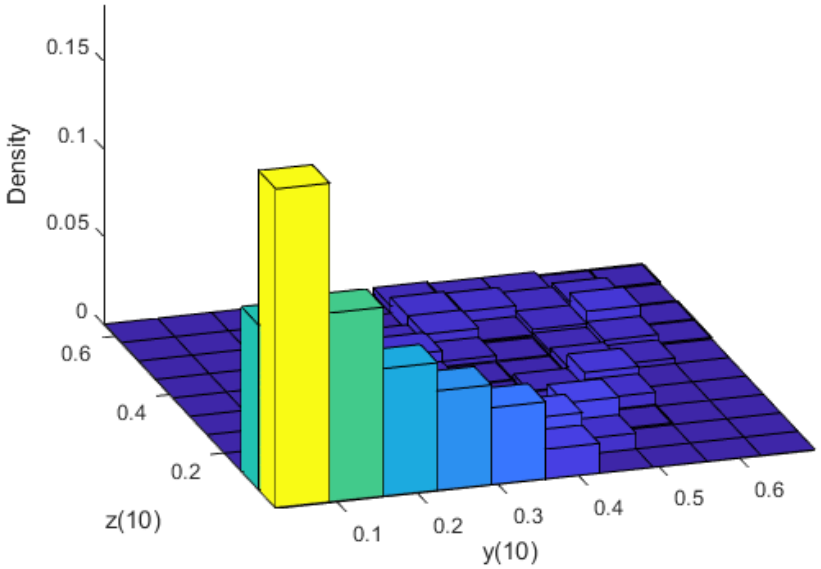
The monteCarloQ2.m function (appendix section 5.8) is similar to that used in Q1b. It does not need to generate the initial conditions as we did in that question, and it uses the Euler-Maruyama approximations instead of the forward Euler ones.

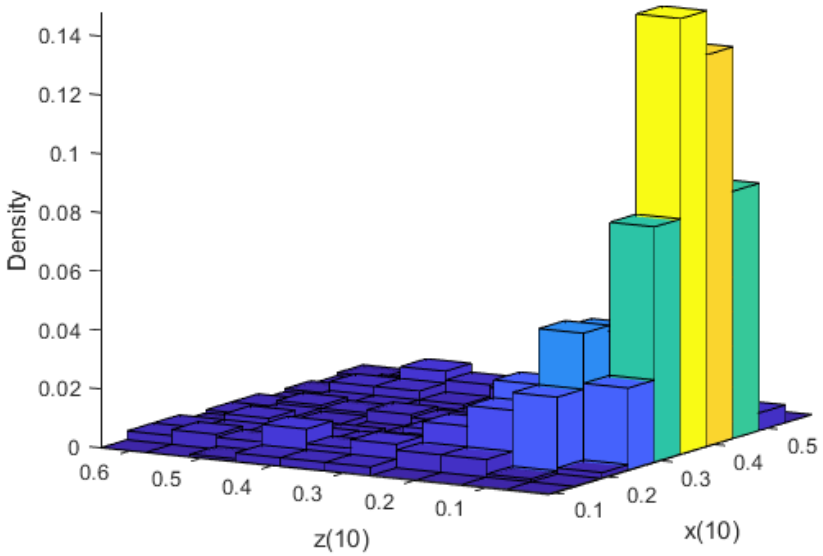
4.2 Distribution Plots and MC Estimates

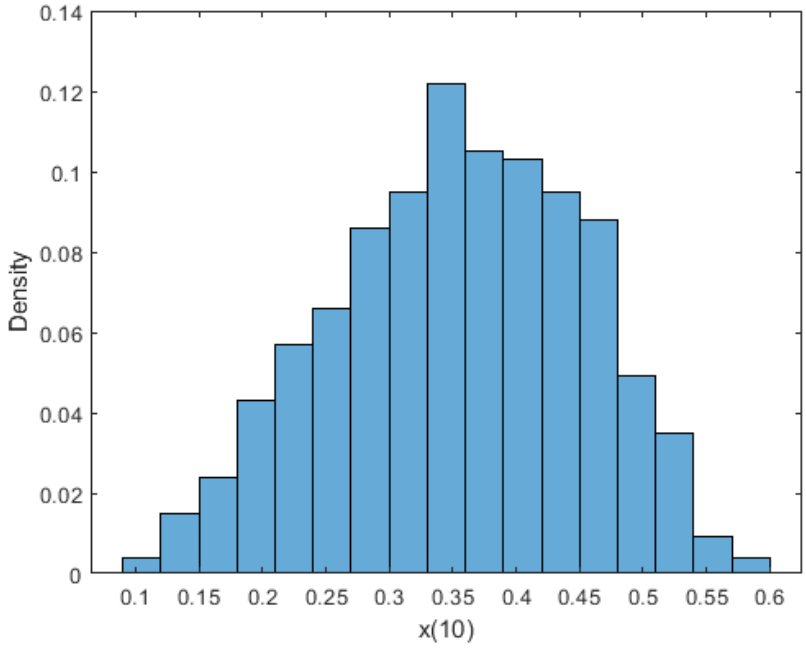
The code in appendix section 5.9 uses $N = 1000$ observations to find the MC estimates. Due to the random nature of the SDE, these estimates change each time we run it. They sit around 0.353, 0.203 and 0.137 respectively. We could get a more accurate estimate if we used more observations N, but we stick with $N = 1000$ for our purposes here.

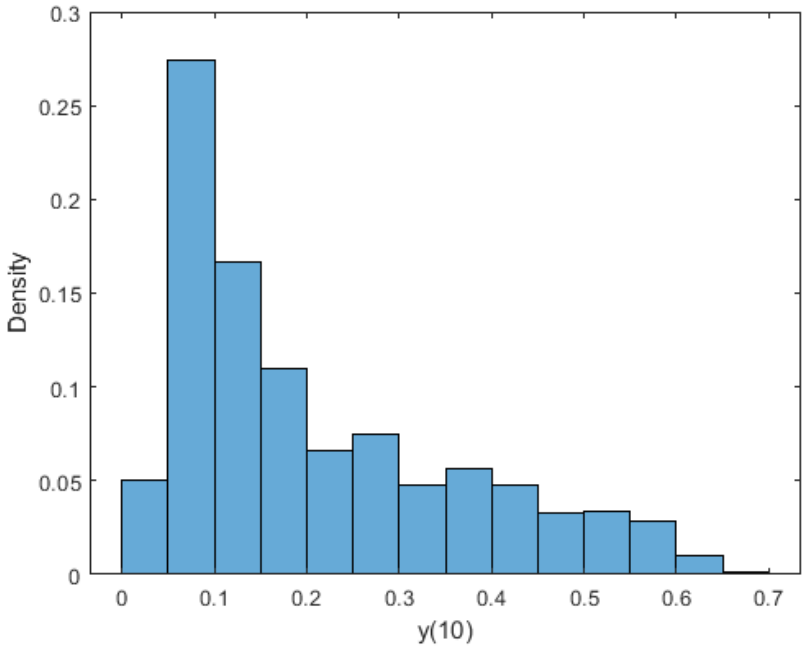
The code then plots the joint distributions between X_{10}, Y_{10} and Z_{10} in pairs, as well as their marginal distributions.

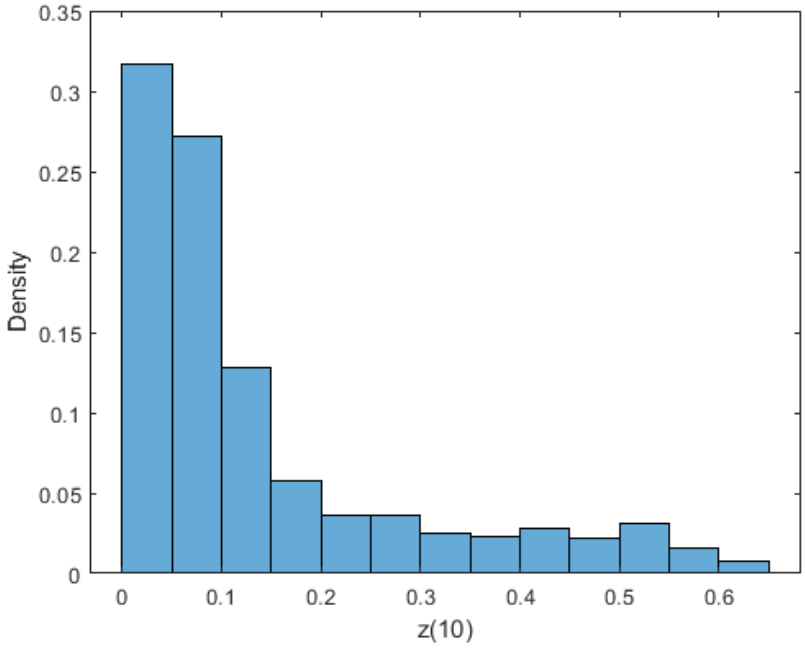

Density
0.1
0.08
0.06
0.04
0.02
0
0.5
0.4
0.3
0.2
0.1
x(10)
0.6
0.5
0.4
0.3
0.2
0.1
y(10)


Density
0.15
0.1
0.05
0
0.6
0.4
0.2
z(10)
0.1
0.2
0.3
0.4
0.5
0.6
y(10)


Density
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.6
0.5
0.4
0.3
0.2
0.1
z(10)
0.1
0.2
0.3
0.4
0.5
x(10)


Density
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
x(10)


Density
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
y(10)


Density
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.1
0.2
0.3
0.4
0.5
0.6
z(10)

4.3 Standard Error

The argument for keeping the standard error below 10^{-3} is identical to that given in Q1b. Please read above for the justification.

The only difference is that we will get different values for the sample variances. These are again calculated using the same code in appendix section 5.5. With $n = 1000$, we get sample variances 0.1345, 0.0669, 0.0405 for X, Y and Z respectively. Note that due to the random nature of the SDE, these values will vary.

If we look at just X for now, we can calculate a lower bound on N, using $\hat{V}_n = 0.1345$:

$$s.e.(X_N) \approx \sqrt{\frac{1}{N}\hat{V}_n} \leq 10^{-3}$$

$$\Rightarrow N \geq 10^6 \hat{V}_n \approx 134,500$$

after some rearranging.

Similarly for Y and Z, we get $66,900$ and $40,500$ as the lower bounds for N. It is important that these are rough estimations, due to the randomness present throughout the calculations.

5 Appendix

I attach the MATLAB code used throughout this coursework, including functions and general scripts.

5.1 forwardEuler.m

```
%% forwardEuler function

function [X,Y,Z] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,
    T,dt,X0,Y0,Z0)
% a1,a2,a3,b1,b2,b3,K,n are the constants determined
    beforehand
% T is the final time
% dt is the timestep length
% X0,Y0,Z0 is the initial condition

N = T/dt;     % number of required time steps
N = int64(N);     % convert from standard form to
    integer before zeros function
X = zeros(N+1,1);     % array to hold values of X
Y = zeros(N+1,1);     % array to hold values of Y
Z = zeros(N+1,1);     % array to hold values of Z
```

```
% set the initial conditions
X(1) = X0;
Y(1) = Y0;
Z(1) = Z0;

% for each timestep, apply the explicit Euler approx
   to X,Y and Z
for i = 1:N
    X(i+1) = X(i) + dt * (a1 - (b1*X(i)*Z(i)^n)/(K^n +
        Z(i)^n));
    Y(i+1) = Y(i) + dt * ((a2*(1-Y(i))*X(i)^n)/(K^n+X(
       i)^n) - b2*Y(i));
    Z(i+1) = Z(i) + dt * ((a3*(1-Z(i))*Y(i)^n)/(K^n+Y(
       i)^n) - b3*Z(i));
end

end
```

5.2 Question 1a

```
%% Question 1a

% define all the variables
a1=0.1;
a2=3;
a3=3;
b1=3;
b2=1;
b3=1;
K=0.5;
n=8;
T=10;
dt=1e-5;
X0=0.1;
Y0=3;
Z0=3;

% run the forward Euler algorithm
[X,Y,Z] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt,X0,
   Y0,Z0);

% observe the approximations for x(10),y(10) and z(10)
X(end)
Y(end)
Z(end)
```

```
% define the ticks on the x-axis
t = 0:dt:T;

% plot the approximations for X, Y and Z
figure(1)
plot(t,X,'DisplayName','X')
hold on
plot(t,Y,'DisplayName','Y')
hold on
plot(t,Z,'DisplayName','Z')
xlabel('Time')
ylabel('Approximation')
legend
grid on


% define the new timestep length
dt1=1e-1;
% run the forward Euler algorithm with this new
    timestep length
[X1,Y1,Z1] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt1,
    X0,Y0,Z0);
% calculate the absolute discrepancy between this new
    approx and the
% original approx
abs_disc1 = sqrt((X(end)-X1(end))^2 + (Y(end)-Y1(end))
    ^2 + (Z(end)-Z1(end))^2);

% repeat with all of the other timesteps
dt2 = 1e-2;
[X2,Y2,Z2] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt2,
    X0,Y0,Z0);
abs_disc2 = sqrt((X(end)-X2(end))^2 + (Y(end)-Y2(end))
    ^2 + (Z(end)-Z2(end))^2);

dt3 = 1e-3;
[X3,Y3,Z3] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt3,
    X0,Y0,Z0);
abs_disc3 = sqrt((X(end)-X3(end))^2 + (Y(end)-Y3(end))
    ^2 + (Z(end)-Z3(end))^2);

dt4 = 1e-4;
[X4,Y4,Z4] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt4,
    X0,Y0,Z0);
abs_disc4 = sqrt((X(end)-X4(end))^2 + (Y(end)-Y4(end))
```

```
    ^2 + (Z(end)-Z4(end))^2);

% plot our graph, using log scales for better
    presentability
xvalues = [dt4,dt3,dt2,dt1];
yvalues = [abs_disc4,abs_disc3,abs_disc2,abs_disc1];
figure(2)
loglog(xvalues,yvalues)
xlabel('Timestep length')
ylabel('Absolute discrepancy, compared to dt=1e-5')
grid on
```

5.3 monteCarloQ1.m

```
%% Monte Carlo algorithm for Q1b

function [x10s, y10s, z10s, xN, yN, zN] = monteCarloQ1
    (N)
%INPUTS
%N - the number of samples we want to use in our Monte
     Carlo estimation


%OUTPUTS
% x10s - an Nx1 vector, containing each sample of x
    (10)
% y10s - similar
% z10s - similar
% xN - the MC estimate for x(10), using all of these
    samples
% yN - similar
% zN - similar


% define the variables as used in Q1a, except for the
    ICs X0,Y0,Z0
a1=0.1;
a2=3;
a3=3;
b1=3;
b2=1;
b3=1;
K=0.5;
n=8;
T=10;
```

```
dt=1e-2;

% these vectors store each sample of x(10), y(10) and
   z(10)
x10s = zeros(N,1);
y10s = zeros(N,1);
z10s = zeros(N,1);

% these samples are also added to the sum for each MC
   estimate
sumX = 0;
sumY = 0;
sumZ = 0;

% sample x(10) and y(10) N times
for i = 1:N

    % rejective sampling for the ICs
    % give dummy negative values to the ICs to make
       sure we initially enter the while loop
    X0 = -1;
    Y0 = -1;
    Z0 = -1;
    % keep drawing random IC values, until all three
       are positive
    while (X0 < 0) || (Y0 < 0) || (Z0 < 0)
        X0 = normrnd(0.1,0.01);
        Y0 = normrnd(3,0.01);
        Z0 = normrnd(3,0.01);
    end

    % run the Euler approximation function
    [X,Y,Z] = forwardEuler(a1,a2,a3,b1,b2,b3,K,n,T,dt,
       X0,Y0,Z0);
    % X(end), Y(end) and Z(end) here give one sample
       each of x(10), y(10)and z(10)
    % save these to our vectors
    x10s(i) = X(end);
    y10s(i) = Y(end);
    z10s(i) = Z(end);
    % also add them to our sums
    sumX = sumX + X(end);
    sumY = sumY + Y(end);
    sumZ = sumZ + Z(end);
end
```

```
% divide our sums by the number of samples N for our
    MC estimates
xN = sumX/N;
yN = sumY/N;
zN = sumZ/N;

end
```

5.4 Question 1b

```
%% Question 1b

N = 1000;
% run the MC algorithm with N = 1000 samples, for
    example
[x10s, y10s, z10s, xN, yN, zN] = monteCarloQ1(N);
% increasing N reduces bias of the estimators, but
    increases running time

% our estimates for E(x(10)), E(y(10)), E(z(10)) with
    N = 1000, dt = 10^-2
xN
yN
zN

% visualise the joint distribution of x(10) and y(10)
figure(1)
hist2d(x10s, y10s, 'normalization', 'probability')
xlabel('x(10)')
ylabel('y(10)')
zlabel('Density')

% visualise the marginal distributions of x(10) and y
    (10)
figure(2)
histogram(x10s,'normalization','probability')
xlabel('x(10)')
ylabel('Density')

figure(3)
histogram(y10s,'normalization','probability')
xlabel('y(10)')
ylabel('Density')
```

5.5 Calculating sample variances

```
%% Sample variance calculations
% used in the 'standard error' parts of Q1b and Q2b

% calculating the sample variance, with n = 1000,
   since we have already
% taken samples from the relevant distribution!
sum1 = 0;
sum2 = 0;
for j = 1:N
    sum1 = sum1 + x10s(j)^2;
    sum2 = sum2 + x10s(j);
end
sample_variance_x = (1/(N-1))*(sum1 - ((1/N) * sum2)
   ^2)

% repeat with y and z
sum1 = 0;
sum2 = 0;
for j = 1:N
    sum1 = sum1 + y10s(j)^2;
    sum2 = sum2 + y10s(j);
end
sample_variance_y = (1/(N-1))*(sum1 - ((1/N) * sum2)
   ^2)

sum1 = 0;
sum2 = 0;
for j = 1:N
    sum1 = sum1 + z10s(j)^2;
    sum2 = sum2 + z10s(j);
end
sample_variance_z = (1/(N-1))*(sum1 - ((1/N) * sum2)
   ^2)
```

5.6 eulerMaruyama.m

```
%% eulerMaruyama function

function [X,Y,Z] = eulerMaruyama(gamma,a1,a2,a3,b1,b2,
   b3,K,n,T,dt,X0,Y0,Z0)
% gamma,a1,a2,a3,b1,b2,b3,K,n are the constants
   determined beforehand
% T is the final time
```

```
% dt is the timestep length
% X0,Y0,Z0 is the initial condition

N = T/dt;    % number of required time steps
N = int64(N);    % convert from standard form to
   integer before zeros function
X = zeros(N+1,1);    % array to hold values of X
Y = zeros(N+1,1);    % array to hold values of Y
Z = zeros(N+1,1);    % array to hold values of Z

% set the initial conditions
X(1) = X0;
Y(1) = Y0;
Z(1) = Z0;

% for each timestep, draw omega from N(0,1), then
   apply the E-M approximation formulas
for i = 1:N
    omega = randn;
    X(i+1) = X(i) + dt * (a1 - (b1*X(i)*Z(i)^n)/(K^n +
        Z(i)^n)) + gamma*X(i)*sqrt(dt)*omega;
    Y(i+1) = Y(i) + dt * ((a2*(1-Y(i))*X(i)^n)/(K^n+X(
       i)^n) - b2*Y(i)) + gamma*Y(i)*sqrt(dt)*omega;
    Z(i+1) = Z(i) + dt * ((a3*(1-Z(i))*Y(i)^n)/(K^n+Y(
       i)^n) - b3*Z(i)) + gamma*Z(i)*sqrt(dt)*omega;
end

end
```

5.7 Question 2a

```
%% Question 2a
% define all the variables
gamma=0.1;
a1=0.1;
a2=3;
a3=3;
b1=3;
b2=1;
b3=1;
K=0.5;
n=8;
T=10;
dt=1e-2;
X0=0.3;
```

```
Y0=0.1;
Z0=0.1;

% define the ticks on the x-axis
t = 0:dt:T;

% plot how X,Y,Z change with each timestep
% repeat 5 times
for counter=1:5
    [X,Y,Z] = eulerMaruyama(gamma,a1,a2,a3,b1,b2,b3,K,
       n,T,dt,X0,Y0,Z0);
    figure(1)
    plot(t,X)
    hold on
    xlabel('Time')
    ylabel('X approximation')
    grid on

    figure(2)
    plot(t,Y)
    hold on
    xlabel('Time')
    ylabel('Y approximation')
    grid on

    figure(3)
    plot(t,Z)
    hold on
    xlabel('Time')
    ylabel('Z approximation')
    grid on
end
```

5.8 monteCarloQ2.m

```
% define the variables as used in Q2a
gamma=0.1;
a1=0.1;
a2=3;
a3=3;
b1=3;
b2=1;
b3=1;
K=0.5;
n=8;
```

```
T=10;
dt=1e-2;
X0=0.3;
Y0=0.1;
Z0=0.1;

% these vectors store each sample of x(10), y(10) and
   z(10)
x10s = zeros(N,1);
y10s = zeros(N,1);
z10s = zeros(N,1);

% these samples are also added to the sum for each MC
   estimate
sumX = 0;
sumY = 0;
sumZ = 0;

% sample x(10), y(10) and z(10) N times
for i = 1:N
    % run the eulerMaruyama approximation function
    [X,Y,Z] = eulerMaruyama(gamma,a1,a2,a3,b1,b2,b3,K,
       n,T,dt,X0,Y0,Z0);
    % X(end), Y(end) and Z(end) here give one sample
       each of x(10), y(10)and z(10)
    % save these to our vectors
    x10s(i) = X(end);
    y10s(i) = Y(end);
    z10s(i) = Z(end);
    % also add them to our sums
    sumX = sumX + X(end);
    sumY = sumY + Y(end);
    sumZ = sumZ + Z(end);
end

% divide our sums by the number of samples N for our
   MC estimates
xN = sumX/N;
yN = sumY/N;
zN = sumZ/N;

end
```

5.9 Question 2b

```
%% Question 2b

N = 1000;
% run the MC algorithm with N = 1000 samples, for
    example
[x10s, y10s, z10s, xN, yN, zN] = monteCarloQ2(N);
% increasing N reduces bias of the estimators, but
    increases running time

% our estimates for E(x(10)), E(y(10)), E(z(10)) with
    N = 1000, dt = 10^-2
xN
yN
zN

% visualise the joint distribution of x(10) and y(10)
figure(1)
hist2d(x10s, y10s, 'normalization', 'probability')
xlabel('x(10)')
ylabel('y(10)')
zlabel('Density')

% y(10) and z(10)
figure(2)
hist2d(y10s, z10s, 'normalization', 'probability')
xlabel('y(10)')
ylabel('z(10)')
zlabel('Density')

% x(10) and z(10)
figure(3)
hist2d(x10s, z10s, 'normalization', 'probability')
xlabel('x(10)')
ylabel('z(10)')
zlabel('Density')

% visualise the marginal distributions of x(10)
figure(4)
histogram(x10s,'normalization','probability')
xlabel('x(10)')
ylabel('Density')

% y(10)
figure(5)
histogram(y10s,'normalization','probability')
xlabel('y(10)')
```

```
ylabel('Density')

% z(10)
figure(6)
histogram(z10s,'normalization','probability')
xlabel('z(10)')
ylabel('Density')
```