Population Dynamics

Jacob Hines-Den (730010228), Joseph Hogsdon (730023668), Mikhail Rodko (730059618), Amy McCulloch (730025794), Cole Mitchell (710033943), Eleanor Ward (730005601)

Introduction

Considering the population of two species of fish: Exe-halibuts (predators) and Wye-bait (prey), we modelled their populations using the Lotka-Volterra model with Matlab. Their respective populations are x(t) and y(t), in suitably scaled units.

1 Conservation

Lotka-Volterra model (System (1)):

$$\dot{x} = x(-1 + y)$$
$$\dot{y} = y(4 - 8x)$$

In system 1 of the Lotka Volterra model, we determined the equilibrium points to be (0,0) and ($\frac{1}{2}$,1). By constructing a Jacobian matrix and employing the eigenvalue equation at either equilibrium point, we classified the points as saddle (unstable) and centre (stable), respectively.

Jacobian Matrix:

$$\begin{pmatrix} -1 + y & x \\ -8y & 4 - 8x \end{pmatrix}$$

Next, we calculated the conserved quantity for this system. Using variable separation and integration, we were able to derive the following:

$$C = y - \ln y - 4 \ln x + 8x$$

In order to determine the possible range of values for E(x,y), we represented x and y in matrix form with the following intervals: x [0,2], y [0,4] and executed the code with 0.001 and 0.002 spacing for x and y, respectively. As a result, we chose to plot the conserved quantity for E(x,y) [-16,17].

Figure 1 illustrates the equilibrium points. The trajectories, represented by the contour lines, are elliptical around the stable centre point ($\frac{1}{2}$, 1) and hyperbolic around the unstable saddle point (0, 0).

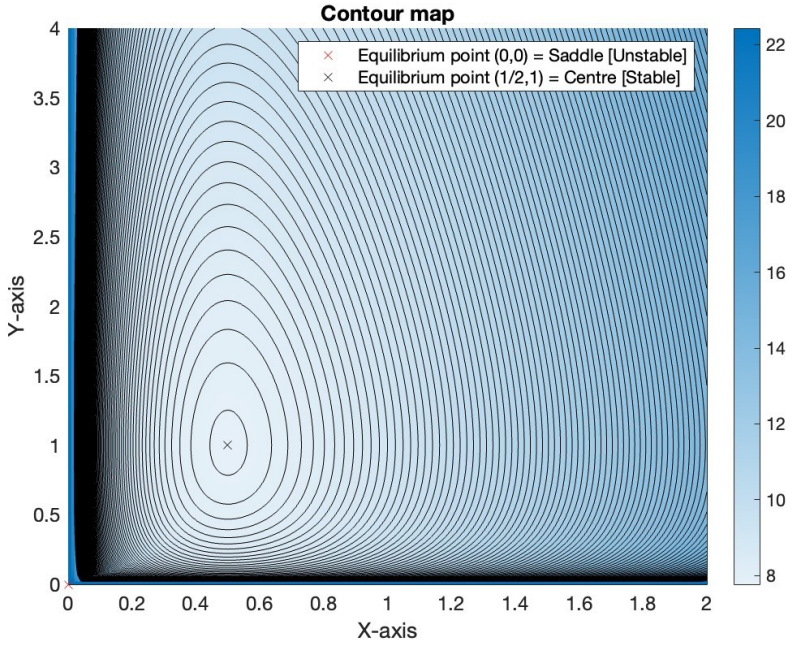

Figure 1: Contour Map

Subsequently, we used the Forward Euler time integration scheme to find the solution to (system 1) for t [0, 5] and initial conditions (x(0), y(0)) = (0.75, 0.5). We took a time step of h=0.1.

$$x(t_{n+1}) = x(t_n) + h(x(t_n)y(t_n) - x(t_n))$$
$$y(t_{n+1}) = y(t_n) + h(4y(t_n) - 8x(t_n)y(t_n))$$

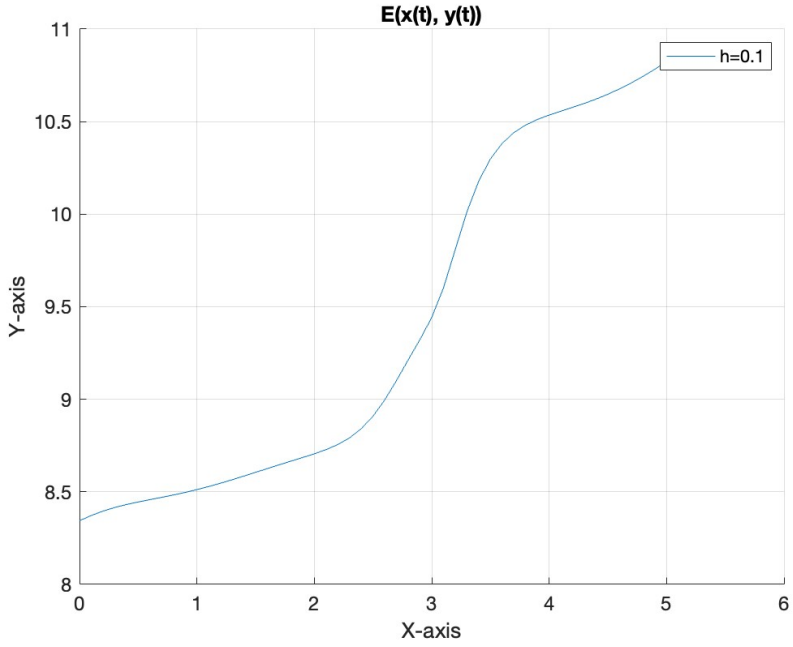

Figure 2: E(x(t), y(t))

Substituting our solution for x(t), y(t) into the expression for E(x,y), we were able to compute the quantity E(t) = E(x(t), y(t)) for this numerical solution, as shown in Figure 2.

After calculating |(E(t) - E(0))/E(0)| for t= 0.1, we found that E increased by approximately 29% at the final time, t=5, indicating that E was not well conserved.

Repeating the calculation for smaller values of △t, we observed that the △t0=0.01 was required to ensure that at the final time t=5, the relative change E, |(E(t) - E(0))/E(0)|, is less than 2%

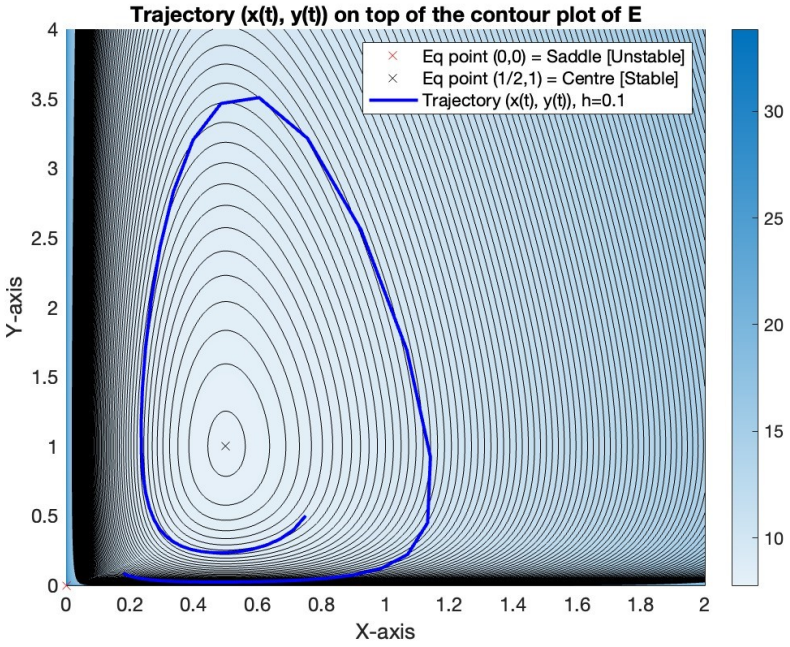

Figure 3: Trajectory on top of the contour plot of E

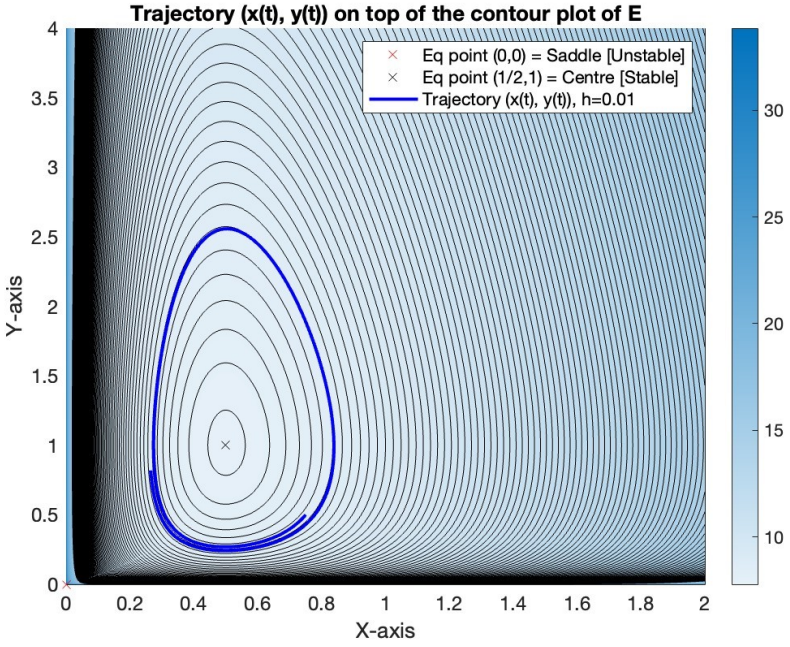

Figure 4: Trajectory on top of the contour plot of E

In Figure 3, while the trajectory of x(t) and y(t) hinted at an elliptical shape with a time step of h = 0.1, it appeared too scattered. This discrepancy with the contour lines, implies that a smaller time step is needed to model the system's behaviour.For h=0.01, however, the trajectory resembled an ellipse, as evidenced by the almost identical mapping onto the contour lines, shown in Figure 4. This is consistent with the theory, as decreasing the time step enhances the accuracy of approximation of the system. Thus, it is reasonable to use this model to model the system at h=0.01.

2 Nullclines and Equilibria

In this section we used the Lotka-Volterra model of

$$\dot{x} = x(-1 - x + y)$$
$$\dot{y} = y(4 - \mu y - 8x)$$

with $\mu = 1$.

We used this model to plot the nullclines of this system and then find and classify the equilibrium points. From the equations, we set $\dot{x}$ **and** $\dot{y}$ **to 0. This gave us either x=0 or y=x+1 from setting** $\dot{x}$ **to 0 and y=0 or y=4-8x from setting** $\dot{y}$ **to 0. These equations allow us to plot the nullcines.**

The equilibrium points are where the $\dot{x}$ **and** $\dot{y}$ **nullcines intersect. So these are (0,0), (0,4), (-1,0) and** $(\frac{1}{3}, \frac{4}{3})$.

We then used the Jacobian matrix to classify the equilibrium points along with the expanded out Lotka-Volterra model:

$$\begin{pmatrix} F_x & F_y \\ G_x & G_y \end{pmatrix}$$

$$\dot{x} = -x - x^2 + xy = F(x, y)$$
$$\dot{y} = 4y - y^2 - 8xy = G(x, y)$$

With this, we calculated the partial derivatives to give us the following matrix. This enabled us to plug in our equilibrium points to classify them.

$$\begin{pmatrix} -1 - 2x + y & x \\ -8y & 4 - 2y - 8x \end{pmatrix}$$

After plugging in each equilibrium point into this matrix and then calculating the eigenvalues, we obtained the following classifications.

For the equilibrium point (0,0), the eigenvalues were $\lambda = -1$ **and** $\lambda = 4$. **As both values are real and there's one positive and one negative, this equilibrium point is a saddle point which is unstable.For the equilibrium point (0,4), the eigenvalues were** $\lambda = 3$ **and** $\lambda = -4$. **Again, as both values are real and there's one positive and one negative, this equilibrium point is a saddle point which is unstable. For the equilibrium point (-1,0), the eigenvalues were** $\lambda = 1$ **and** $\lambda = 12$. **As both values are real, positive and not equal, this equilibrium point is an improper unstable node. For the equilibrium point** $(\frac{1}{3}, \frac{4}{3})$, **the eigenvalues were** $\lambda = \frac{-5+\sqrt{119}i}{6}$ **and** $\lambda = \frac{-5-\sqrt{119}i}{6}$. **As these values are complex conjugates with equal and negative real parts, this equilibrium point is a stable spiral.**

Figure 5 below shows the plotted nullclines and the equilibrium points.

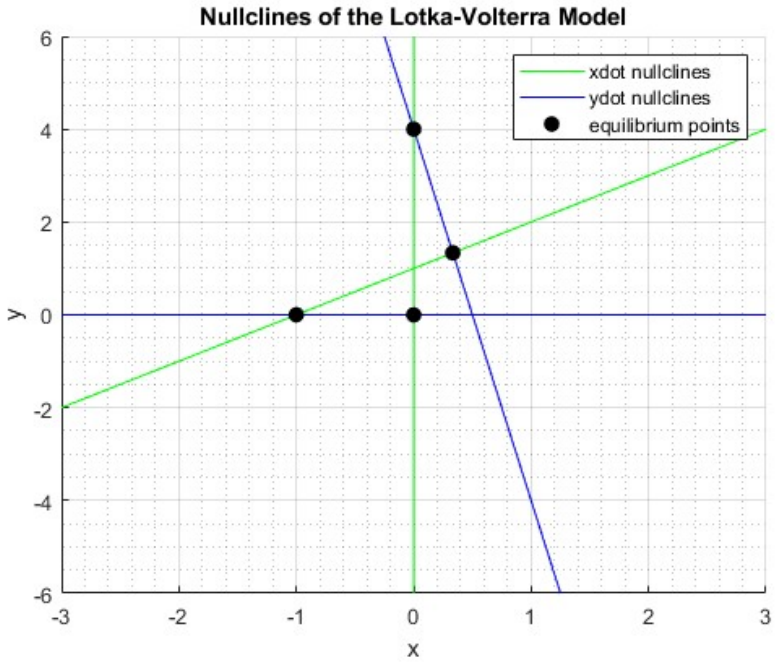

Figure 5: Nullclines of the Lotka-Volterra Model

3 Unstable and Stable Manifolds

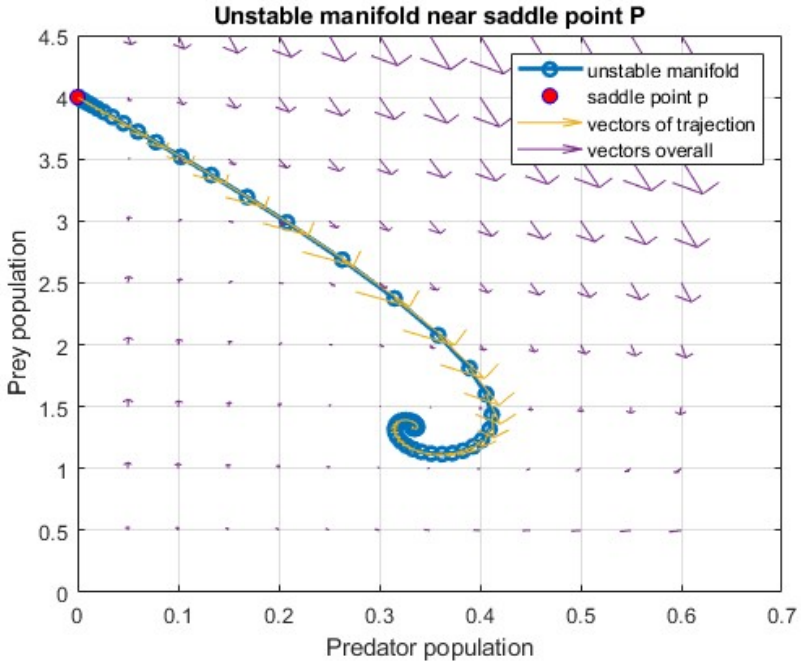

Figure 6: Unstable Manifold Near Saddle Point

The Lotka-Volterra model from section 2 has a saddle point which isn't the origin of (0,4). We then plotted an estimate of the shape of the unstable manifold of this saddle by computing a trajectory that starts very close to the saddle point. The initial conditions we used as close to the saddle point were x0 = 0.001 and y0 = 3.9999. Using the equations given in the Lotka-Volterra model in section 2 and Matlab, we solved the ODE which allowed us to plot this graph shown in figure 6. We also included quivers to show trajection.

To find the location of the stable manifold of the saddle, we used the found Jacobian matrix of the equilibrium point (0,4) with our eigenvalue at (0,4) of $\lambda = -4$.

$$\begin{pmatrix} 3 & 0 \\ -32 & -4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 3-(-4) & 0 \\ -32 & -4-(-4) \end{pmatrix} = \begin{pmatrix} 7 & 0 \\ -32 & 0 \end{pmatrix}$$

We know the direction follows the eigenvectors which come from:

$$\begin{pmatrix} 7 & 0 \\ -32 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

This then gave us the eigenvector of $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. So the stable manifold is straight along the y-axis, just above (0,0).

We then looked at the direction of the unstable manifold near the saddle point (0,4). For this we used the eigenvalue at (0,4) of $\lambda = 3$ and used the same method as with the stable manifold, giving us the eigenvector of $\begin{pmatrix} 1 \\ \frac{-32}{7} \end{pmatrix}$.

We then determined empirically the direction of the unstable manifold near the saddle point, by estimating dy/dx of our numerical solution. We took 2 pairs of values close to the saddle from the matlab graph in figure 6 and calculated change in y over change in x. This estimate of dy/dx gave us -4.6868.The direction predicted by linear analysis above gave $\frac{-32}{7} = -4.5714$. Therefore we can say that the direction agrees as they are close in value.

4 Maximum Sustainable Yield

4.1 Finding C_{MSY}

Maximum sustainable yield is defined in fisheries as the maximum catch that can be removed from a population over an indefinite period [1]. In this section we are determining the maximum value of C (the maximum sustainable yield) for which there exists a stable equilibrium.

Humans wish to catch Exe-halibuts for food. Suppose, for simplicity, they do so at a constant rate C 0, so that the Lotka–Volterra model becomes:

$$\dot{x} = x(-1 - x + y) - C$$

$$\dot{y} = y(4 - \mu y - 8x)$$

and for this task we fixed $\mu = 2$. Solving the system for equilibrium points we acquired the following x-values: $x_{1,2} = \frac{-1 \pm \sqrt{1-4C}}{2}$ and $x_{3,4} = \frac{1 \pm \sqrt{1-20C}}{10}$. Using these values, we were able to create inequalities for C that encompassed all possible equilibrium points.

$$C = \frac{1}{4} \longrightarrow C < \frac{1}{4}, C = \frac{1}{20} \longrightarrow C < \frac{1}{20}$$

This is best illustrated below in figure 7:

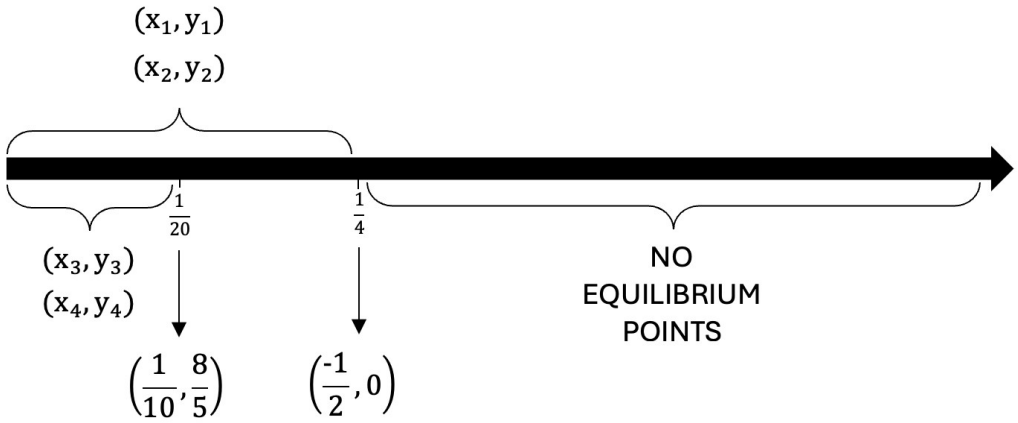

Figure 7: Diagram to show inequalities for C

By constructing a Jacobian matrix and utilising it to compute the equations for △ = (ad-bc) and trace = (a+b), we proceeded to find the maximum value of C for which there exists a stable equilibrium with a nonzero population of Exe-halibuts.

$$A = \begin{pmatrix} \dot{x}_x & \dot{x}_y \\ \dot{y}_x & \dot{y}_y \end{pmatrix} = \begin{pmatrix} -1-2x+y & x \\ -8y & 4-4y-8x \end{pmatrix}$$

$$\triangle = (-1-2x+y)(4-4y-8x) + 8xy$$

$$Trace = -1-2x+y+4-4y-8x = 3-3y-10x$$

Through exhaustive proof at each interval of C, we found that a stable equilibrium point existed at (x_3, y_3) where $\frac{-3}{4} < C < \frac{1}{20}$. Evaluating △ and trace at $C = \frac{1}{20}$, where the equilibrium point is $(\frac{1}{10}, \frac{8}{5})$, we found that $\triangle = 0$ and $trace = \frac{-14}{5}$

$$x_3 = \frac{1}{10} + \frac{\sqrt{1-20C}}{10}, y_3 = \frac{8}{5} - \frac{2\sqrt{1-20C}}{5}$$

Investigating further, via the formula

$$\lambda = \frac{trace \pm \sqrt{trace^2 - 4\triangle}}{2},$$

we found that the eigenvalues $[\lambda_1, \lambda_2]$ at $C = \frac{1}{20}$, were [0, trace] or [0, $\frac{-14}{5}$], respectively. Although $\lambda_2 = \frac{-14}{5}$ implies stability, additional examination is required for λ_1= 0. The situation at C = $\frac{1}{20}$ is a degenerate case, where two equilibrium points collide and disappear. As a result, (x_3, y_3) is the first computed stable equilibrium point, which means that $C = C_{MSY} = \frac{1}{20}$, where C_{MSY} is the 'maximum sustainable yield'.

4.2 Code

By using Forward Euler, we explored different initial conditions and their respective trajectories to outline the basin of attraction and identify some behaviours of the model.

Initially, we classified the stable equilibrium point by substituting x3, y3 at $C = \frac{9}{10} * \frac{1}{20}$ into the Jacobian matrix and via the eigenvalue equation found lambdas to be approximately -0.39 and 2.33 finding it to be a stable improper node.

Now having the classification and location of the equilibrium point allowed us to start calculating the trajectories for a range of initial conditions. By setting up a grid of different initial conditions and then plotting the ones which converge we were able to outline the Basin of Attraction seen in figure 8.

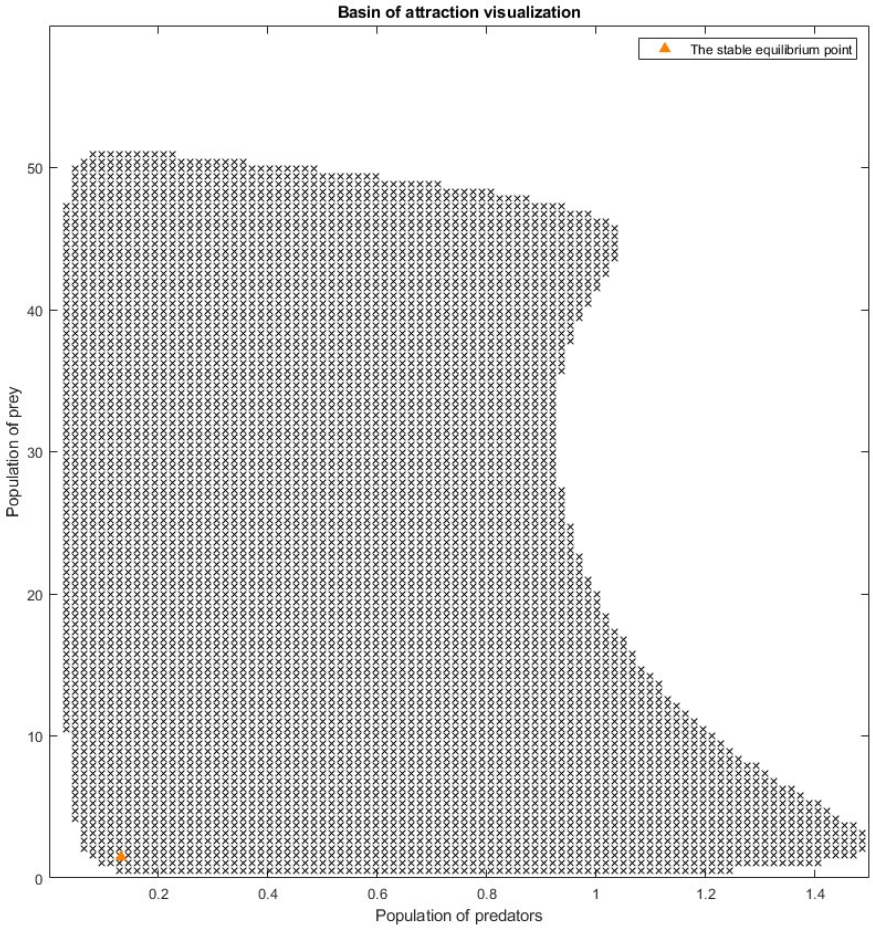

Figure 8: Basin of attraction visualisation

All the trajectories within the basin of attraction converge towards the equilibrium point but doing so while spiralling inwards towards the equilibria as seen in figure 9. This is interesting as it conveys a cyclical nature.

Outside the basin we can observe 3 main scenarios Low Prey High Pred Population, Similar populations and High Prey Low Pred Population. In the Low prey and high predator scenario in figure 10, we observe that generally, the predators die out due to lack of food which allows the prey to make a recovery. Although if the predator's population is large enough the Prey will die out first, but the predators still become extinct.

Figure 11 shows similar populations outside the basin of attraction and we see the predators always become extinct. This is interesting as it shows that the predator population gain isn't majorly affected by the population size of the Prey. This could mean that a large number of prey could be disadvantageous due to the competition between the prey causing them to die out faster.

High prey and low predator populations come within the basin of attraction which could suggest it's the most suitable if you were controlling population sizes.

Overall, the predator population is sensitive, and if not in a small range it has a large threat of extinction. This may be heavily influenced by the prey's population not having a massive

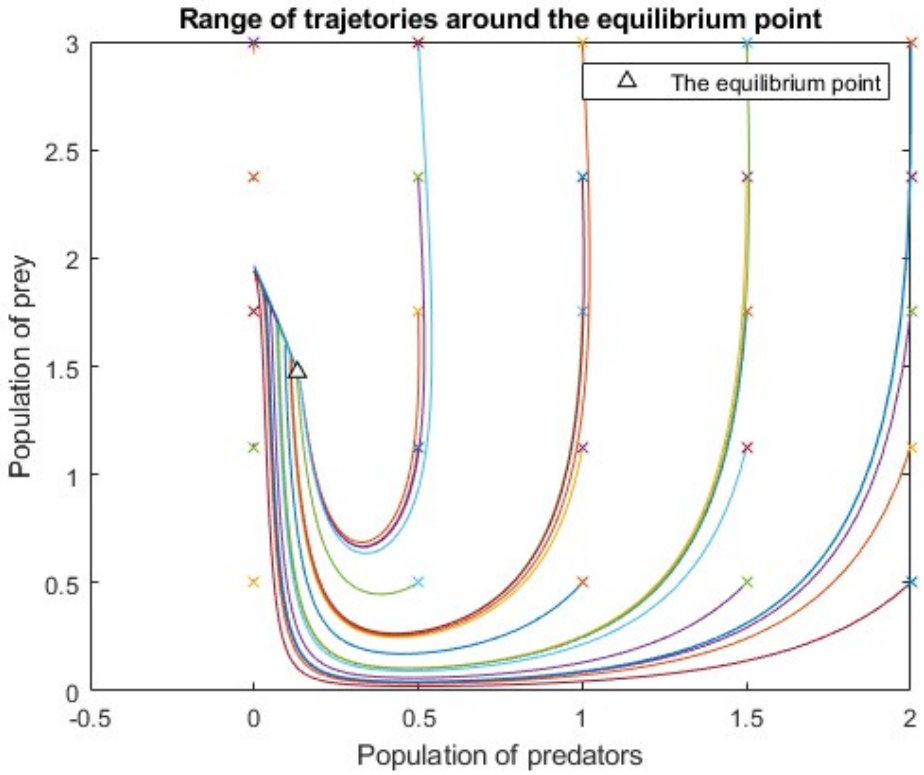

Figure 9: Range of trajectories around the equilibrium point

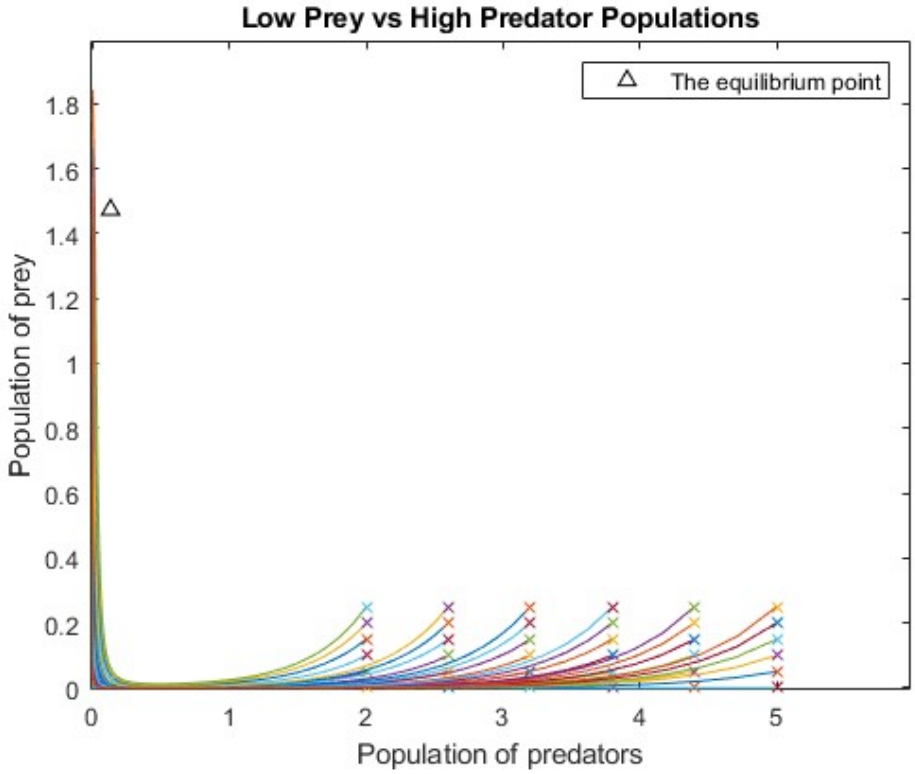

Figure 10: Low Prey vs High Predator

effect on population growth when in excess.

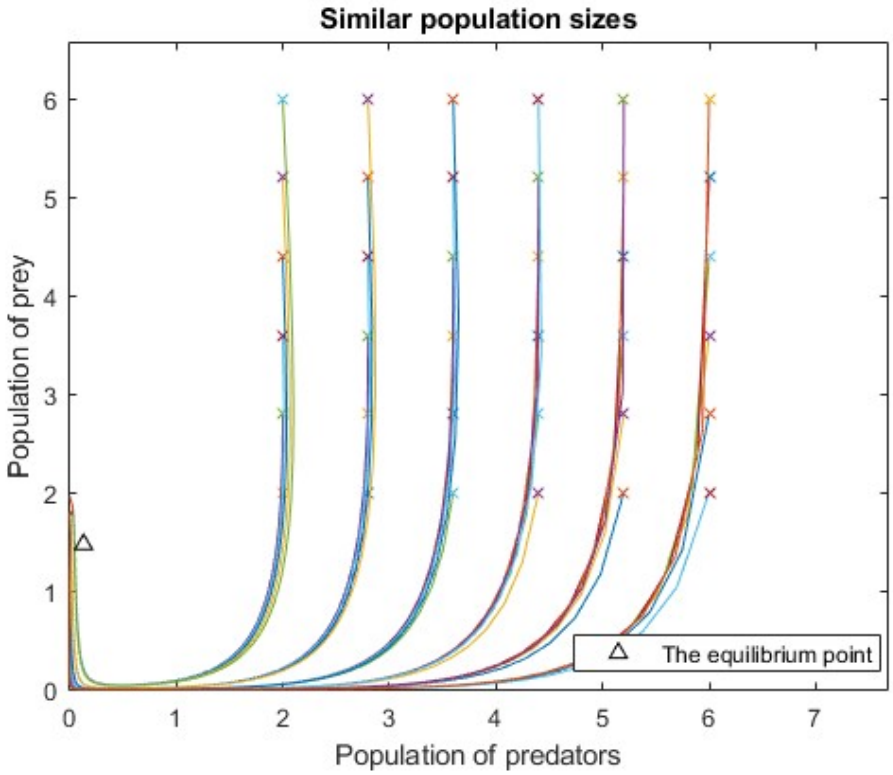

Figure 11: Similar population sizes

4.3 Researching MSY

We then did some background reading on the maximum sustainable yield to help us interpret our results. We found there are multiple limitations to the Maximum Sustainable Yield model as it involves too many assumptions. Often it is difficult to estimate the size of the populations of fish and how fast they are growing. While researching [2], we found that the growth rate is often assumed constant while in reality there are many factors that affect this. For example, there are fluctuations in water conditions like oxygen levels and temperature. Also food web interaction can vary based on other species and sometimes populations are impacted due to parasites. All these factors vary over time, affecting the population and are not considered. This is because the MSY model tends to treat the environment as unvarying and doesn't consider natural fluctuations in population growth.

To make the model more realistic for managing real world species, it should try to model some of these natural fluctuations in the population by studying more in depth the variation of environment conditions which affect them. This will give an estimate of how and when the population size changes, rather than assuming it constant. Then the Maximum Sustainable Yield can be calculated more accurately.

Bibliography

[1] M N Maunder. 2008. ScienceDirect. Maximum Sustainable Yield. Retrieved March 15th 2024 from https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/maximum-sustainable-yield

[2] Stockholm University Baltic Sea Centre. Fact Sheet: Understanding MSY. Retrieved March 19th 2024 from https://www.su.se/stockholm-university-baltic-sea-centre/web-magazine-baltic-eye/fisheries/fact-sheet-understanding-msy

Appendix

Here we have included our code for each section.

Conservation

```
% Contour map:

clear
close all

x = 0:0.01:2 ;
y = 0:0.02:4 ;

[X,Y] = meshgrid(x,y) ;
Z = Y + 8*X - log(abs(Y)) - log(X.^4) ;

cmap = colormap('sky') ;
colormap(cmap) ;
pcolor(X,Y,Z) ;
shading flat ;
colorbar ;
hold on ;
cvals = -16:0.1:17 ;
contour(X,Y,Z,cvals,'k')
p1 = plot(0,0,'rx');
p2 = plot(0.5, 1, 'kx');
legend([p1(1),p2(1)], 'Equilibrium point (0,0) = Saddle [Unstable]',...
    'Equilibrium point (1/2,1) = Centre [Stable]')
hold off


hold on

    xlabel('X-axis')
    ylabel('Y-axis')
    title('Contour map')
    set(gca, 'fontsize', 12)

hold off
```

Figure 12: Code for 1.1

```
clear all

% Initial Conditions:

x(1) = 0.75;
y(1) = 0.5;
t(1) = 0;
E(1) = y(1) + 8*x(1) - log(x(1).^4) - log(abs(y(1)))
h = 0.1
nstop = 51

% Forward Euler:

for n = 1:nstop
    t(n+1) = t(n) + h;
    x(n+1) = x(n) + h*(x(n)*y(n)-x(n));
    y(n+1) = y(n) + h*(4*y(n)-8*x(n)*y(n));
    E(n+1) = y(n+1) + 8*x(n+1) - log(x(n+1).^4) - log(abs(y(n+1)));
end

% Plot of E(t) = E(x(t), y(t))

hold on

    xlabel('X-axis')
    ylabel('Y-axis')
    title('E(x(t), y(t))')
    set(gca, 'fontsize', 12)
    plot(t,E)

hold off

    legend('h=0.1')


% delta t = 0.1 or t = 0.01

abs((E(51) - E(1))/E(1))
```

Figure 13: Code for 1.1

```
% H = 0.1 or 0.01

% Contour map:

x = 0:0.001:2 ;
y = 0:0.002:4 ;

[X,Y] = meshgrid(x,y) ;
Z = Y + 8*X - log(abs(Y)) - log(X.^4) ;

cmap = colormap('sky') ;
colormap(cmap) ;
pcolor(X,Y,Z) ;
shading flat ;
colorbar ;
hold on
cvals = -16:0.1:17 ;
contour(X,Y,Z,cvals,'k')
p1 = plot(0,0,'rx');
p2 = plot(0.5, 1, 'kx');
hold off

% Initial Conditions:

a(1) = 0.75;
b(1) = 0.5;
t(1) = 0;
E(1) = b(1) + 8*a(1) - log(a(1).^4) - log(abs(b(1)))
h = 0.1 % or h = 0.01
nstop = 51 % or nstop = 501

% Forward Euler:

for n = 1:nstop
    t(n+1) = t(n) + h;
    a(n+1) = a(n) + h*(a(n)*b(n)-a(n));
    b(n+1) = b(n) + h*(4*b(n)-8*a(n)*b(n));
    E(n+1) = b(n+1) + 8*a(n+1) - log(a(n+1).^4) - log(abs(b(n+1)));
end
```

Figure 14: Code for 1.1

```
% Plot of trajectory (x(t), y(t)) on top of the contour plot of E. For h =
% 0.1 / 0.01

hold on
    xlabel('X-axis')
    ylabel('Y-axis')
    title('Trajectory (x(t), y(t)) on top of the contour plot of E')
    set(gca, 'fontsize', 12)
p3 = plot(a,b,'LineWidth',2)
set(p3, 'Color', 'b')
hold off

% Change legend for h = 0.1 or h = 0.01

legend([p1(1),p2(1),p3(1)], 'Eq point (0,0) = Saddle [Unstable]',...
    'Eq point (1/2,1) = Centre [Stable]', 'Trajectory (x(t), y(t)), h=0.1')
```

Figure 15: Code for 1.1

Nullclines and Equilibria

```
clear
close all

% setting initial value ranges for xs and ys ready for line equations
x1 = -100 : 100
y2 = -100 : 100
x3 = -100 : 100
y4 = -100 : 100

% line equations
y1 = x1 + 1
x2 = 0.5 - 0.125*y2
y3 = 0 + 0*x3
x4 = 0 + 0*y3


hold on

% plotting each line and giving the xdot and ydot nullclines different
% colours to identify them
plot(x1, y1, 'G')
plot(x2, y2, 'B')
plot(x3, y3, 'B')
plot(x4, y4, 'G')

% plotting each equilibrium point all the same colour and large enough to
% be easily seen
plot(0, 0, '.', 'MarkerSize', 25, 'MarkerEdgeColor', 'black')
plot(0, 4, '.', 'MarkerSize', 25, 'MarkerEdgeColor', 'black')
plot(-1, 0, '.', 'MarkerSize', 25, 'MarkerEdgeColor', 'black')
plot(1/3, 4/3, '.', 'MarkerSize', 25, 'MarkerEdgeColor', 'black')
hold off

% setting up axes and labelling the graph
title('Nullclines of the Lotka-Volterra Model')
xlabel('x')
ylabel('y')
legend('xdot nullclines', 'ydot nullclines', '', '', 'equilibrium points')
xlim([-3,3])
ylim([-6,6])
grid on
grid minor
```

Figure 16: Code for 1.2: Nullclines and Equilibria

Unstable and Stable Manifolds

```
clear
% parameters
u = 1;

% equations
dxdt = @(x, y) x .* (-1 - x + y); % defing functions preditor
dydt = @(x, y) y .* (4 - u * y - 8 * x); %prey

% initial conditions near point P
x0 = 0.001; % inital predator population
y0 = 3.9999;% inital population of prey

% solve ode
timespan = [0 200];
[t, xy] = ode45(@(t, xy) [dxdt(xy(1), xy(2)); dydt(xy(1), xy(2))], timespan, [x0;
y0]);

xp= xy(:,1);
yp= xy(:,2);
xd= xp .* (-1 - xp + yp);
yd= yp .* (4 - u * yp - 8 * xp);

for j = 1:12
    X(j) = 0.05*j;
    for i = 1:9
        Y(i) = 0.5*i;
        xdot(i,j) = X(j)*(-1 - X(j) + Y(i));
        ydot(i,j) = Y(i)*(4 - Y(i) - 8 * X(j));
    end
end


% Plot
figure;
plot(xy(:,1), xy(:,2), 'o-', 'LineWidth', 2);

hold on

plot(0, 4, 'bo', 'MarkerSize', 7, 'MarkerFaceColor', 'r'); % clearity on where inital
point is
quiver(xp,yp,xd,yd,3)
quiver(X,Y,xdot,ydot,1)

xlabel('Predator population');
ylabel('Prey population');
title('Unstable manifold near saddle point P');

legend("unstable manifold", "saddle point p", "vectors of trajection","vectors
overall" );
grid on;
```

Figure 17: Code for 1.3: Unstable and Stable Manifolds

Maximum Sustainable Yield

```
%Clearing the plot and workspace
clf
clear all

%Setting C
c = 0.9 * 0.05;

%Declaring conditions for algorithm
converges = 0;
stepsize = 0.01;
nstop = 5000;
x_equi = 1/10 + ((sqrt(1-(20)*c))/10);
y_equi = 8/5 - (2*(sqrt(1-(20)*c))/5);
%Setting bounds for intitial conditions
x_upper = 1.5;
x_lower = 0.032;

y_upper = 52;
y_lower = 0.53;

%Increments between initial condition bounds
no_increments = 70;
%Creating an empty array
M = zeros(no_increments^2,2);
count = 0;
%Finding the intital conditions
for i = 1:no_increments
    for j = 1:no_increments
        count = count+1;
        M(count,1) = x_lower + (i-1)*(x_upper - x_lower)/(no_increments-1);
        M(count,2) = y_lower + (j-1)*(y_upper - y_lower)/(no_increments-1);
    end
end
```

Figure 18: Code for 1.4: MSY

```
%Creating empty plot
plot(0,0)
title('Basin of attraction visualization')
xlabel('Population of predators')
ylabel('Population of prey')

%Looping throught initial conditions
for i=1:length(M)
    x(1) = M(i,1);
    y(1) = M(i,2);
    t(1) = 0;
    converges = 0;
    %Euler time stepping scheme
    for n = 1:nstop
        t(n+1) = t(n)+ stepsize;
        x(n + 1) = x(n) + stepsize * ( -x(n) -(x(n)*x(n)) + (x(n)*y(n)) - c );
        y(n + 1) = y(n) + stepsize * ((4* y(n)) - (2*y(n)*y(n)) - (8*x(n)*y(n)));
        %If statement to find if the current intital condition converges
        if (norm(x(n + 1) - x_equi) < 1e-8)
            converges = 1;
        end
        %Breaks incase the calculations go below zero
        if(x(n + 1 )<0)
            break
        end
        if(y(n+1)<0)
            break
        end
    end
```

Figure 19: Code for 1.4: MSY

```
    %plotting the trajectory per iteration for each calculation
    hold on
    if (converges == 1)

       plot(M(i,1), M(i,2), 'x', 'Color', 'Black')
    %else

       %plot(M(i,1), M(i,2), 'x', 'Color', 'red')
    end
end
%Plotting stable equilibrium point and

p = plot(x_equi, y_equi, '^');
p.MarkerEdgeColor = [1 0.5 0];
p.MarkerFaceColor = [1 0.5 0];
legend(p,"The stable equilibrium point")
```

Figure 20: Code for 1.4: MSY

```
clear all

%Setting C
c = 0.9 * 0.05;
t(1) = 0;
x(1) = 3;
y(1) = 0.25;
x_equi = 1/10 + ((sqrt(1-(20)*c))/10);
y_equi = 8/5 - (2*(sqrt(1-(20)*c))/5);
stepsize = 0.01;
nstop = 6000;
x_equi = 1/10 + ((sqrt(1-(20)*c))/10);
y_equi = 8/5 - (2*(sqrt(1-(20)*c))/5);
converges = 0;

for n = 1:nstop
        t(n+1) = t(n)+ stepsize;
        x(n + 1) = x(n) + stepsize * ( -x(n) -(x(n)*x(n)) + (x(n)*y(n)) - c );
        y(n + 1) = y(n) + stepsize * ((4* y(n)) - (2*y(n)*y(n)) - (8*x(n)*y(n)));

        if (norm(x(n + 1) - x_equi) < 1e-8)
            converges = 1;
        end
        %Breaks incase the calculations go below zero
        if(x(n + 1 )<0)
            break
        end
        if(y(n+1)<0)
            break
        end

end


if converges == 1
    plot(x,y, 'Color','Blue')
else
    plot(x, y, 'Color','Red')
end
title('Example trajectories')
xlabel('Population of predators')
ylabel('Population of prey')
hold on
p = plot(x_equi, y_equi, '^');
p.MarkerEdgeColor = [0 0 0];
p.MarkerFaceColor = [1 1 1];
legend(p, "The equilibrium point")
```

Figure 21: Code for 1.4: MSY

```
%Clearing the plot and workspace
clf
clear all

%Setting C
c = 0.9 * 0.05;

%Declaring conditions for algorithm

stepsize = 0.01;
nstop = 5000;

x_equi = 1/10 + ((sqrt(1-(20)*c))/10);
y_equi = 8/5 - (2*(sqrt(1-(20)*c))/5);
%Setting bounds for intitial conditions
x_upper = 0.5;
x_lower = 0;

y_upper = 6;
y_lower = 2;

%Increments between initial condition bounds
no_increments = 6;
%Creating an empty array
M = zeros(no_increments^2,2);
count = 0;
%Finding the intital conditions
for i = 1:no_increments
    for j = 1:no_increments
        count = count+1;
        M(count,1) = x_lower + (i-1)*(x_upper - x_lower)/(no_increments-1);
        M(count,2) = y_lower + (j-1)*(y_upper - y_lower)/(no_increments-1);
    end
end
```

Figure 22: Code for 1.4: MSY

```
%Creating empty plot
plot(0,0)
title('Range of trajetories around the equilibrium point')
xlabel('Population of predators')
ylabel('Population of prey')

%Looping throught initial conditions
for i=1:length(M)
    x(1) = M(i,1);
    y(1) = M(i,2);
    t(1) = 0;

    %Euler time stepping scheme
    for n = 1:nstop
        t(n+1) = t(n)+ stepsize;
        x(n + 1) = x(n) + stepsize * ( -x(n) -(x(n)*x(n)) + (x(n)*y(n)) - c );
        y(n + 1) = y(n) + stepsize * ((4* y(n)) - (2*y(n)*y(n)) - (8*x(n)*y(n)));


        %Breaks incase the calculations go below zero
        if(x(n + 1 )<0)
            break
        end
        if(y(n+1)<0)
            break
        end
    end
    %plotting the trajectory per iteration for each calculation
    hold on
    plot(x,y)
    a = plot(M(i,1), M(i,2), 'x');
```

Figure 23: Code for 1.4: MSY