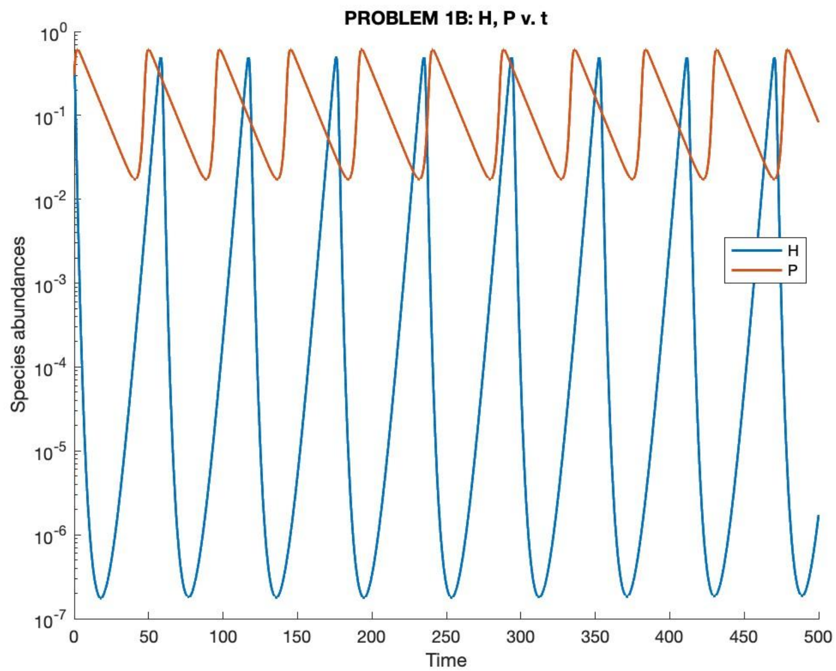
Robert Heeter
BIOE 446
27 October 2023

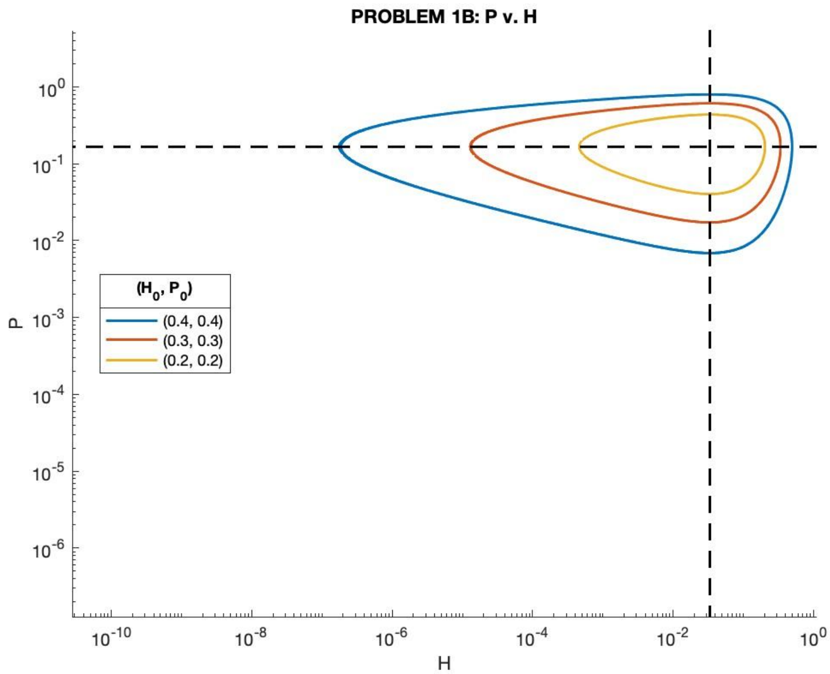
Lab 9: Predator-Prey Dynamics

ANSWERS

1. **Simulation of Lotka-Volterra model (DEMO)**
   a. dH/dt = -0.2800 cells/mL/day
      dP/dt = 0.4400 cells/mL/day

   b. Figures





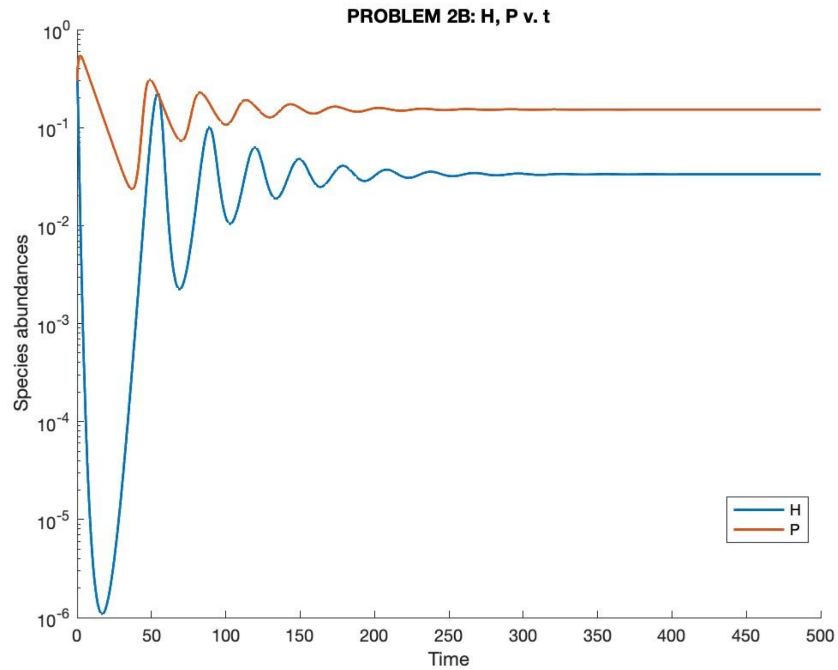
The system never reaches the intersection of the nullclines (indicative of steady-state) regardless of the initial conditions (initial populations). This means that the predator and prey populations never reach a stable equilibrium; they oscillate/fluctuate with a constant amplitude as shown in the first figure, with the predator population (P) always lagging behind the prey population (H).

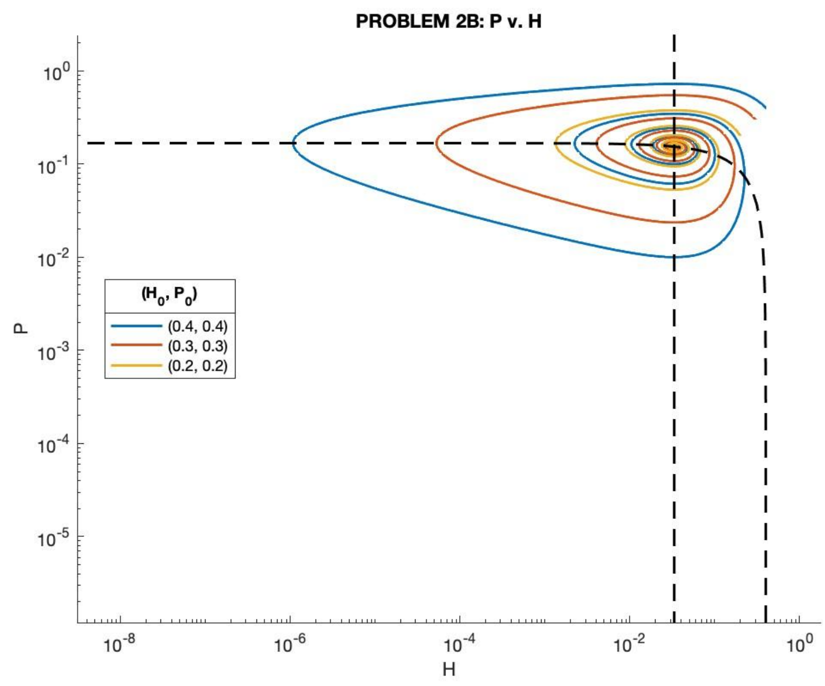
c. *The initial conditions (initial predator and prey populations) only affect the 'amplitude' of the predator and prey abundances in this model. A larger initial population of both predators and prey will result in larger amplitude oscillations in both populations over time. With a larger abundance of both, the predator and prey populations can 'swing' further in a given cycle; a greater number of predators will result in a larger decrease in prey, which in turn will result in a larger decrease in predators, which will result in a larger increase in prey, which will finally result in a larger increase in predators (repeat).*

   The system does not reach a steady state in the long term, which is not realistic since many other factors affect population dynamics. For example, any given environment cannot support an infinitely large population of predators and prey; finite resources result in a 'carrying capacity' for both populations. Furthermore, random events and interactions with other species will obviously impact both predator and prey populations in the long run.

2. **Introducing a carrying capacity on prey growth**
   a. dH/dt = -0.4800 cells/mL/day
   dP/dt = 0.4400 cells/mL/day

b. Figures

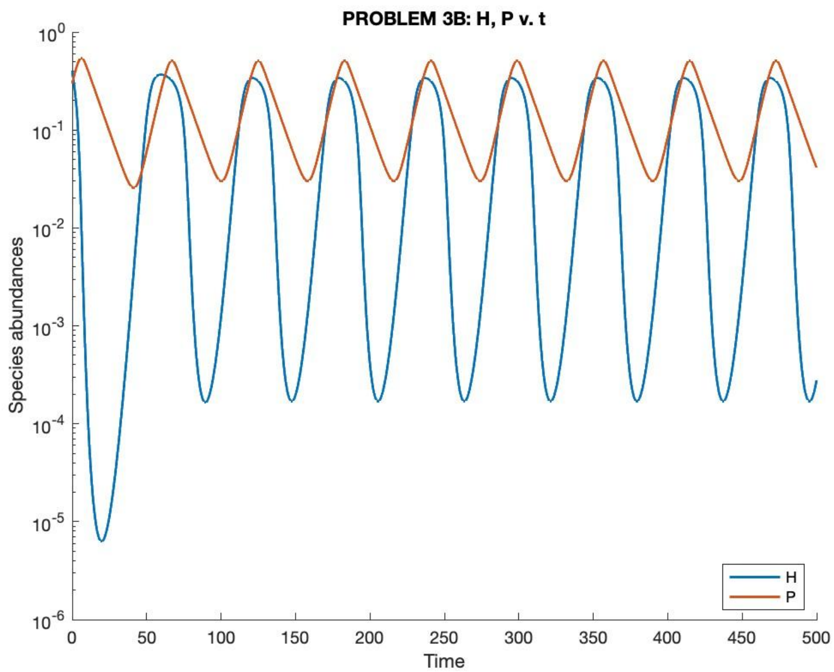


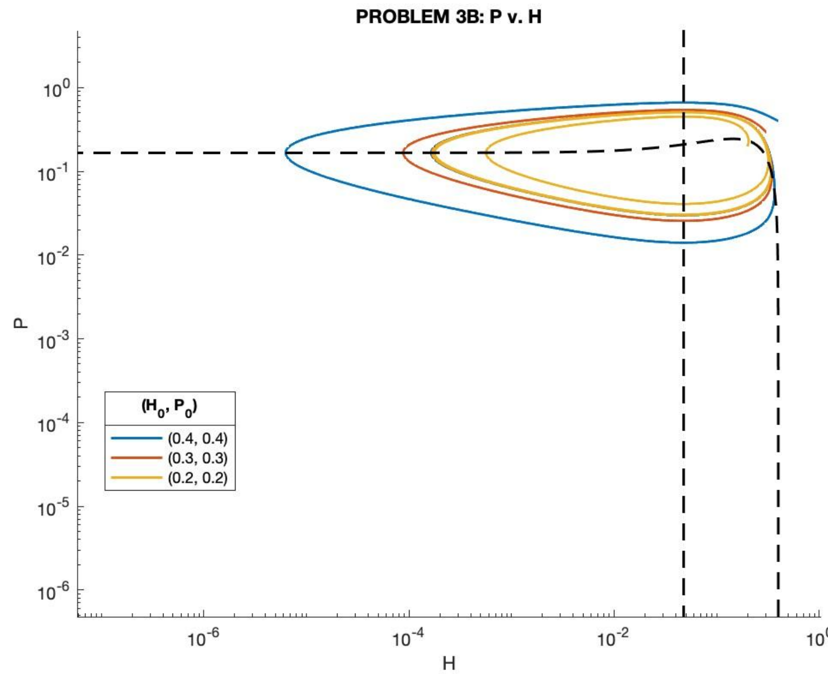


This model with a carrying capacity parameter is more realistic than the basic Lotka-Volterra model in the first problem. Here, the addition of the carrying capacity (representative of limited environmental resources) causes the system to approach an equilibrium/steady-state where both predator and prey populations are stable. This is seen in *both the first figure where oscillations in both populations decay and in the second figure where the populations gravitate towards the intersection of the nullclines (the steady state) regardless of the initial predator and prey populations. This model is still not completely realistic since the predator and prey populations remain at fixed values in the long-run; some reasonable oscillations would be expected.*

3. **Simulating a Macarthur-Rozenzweig model**
    a. dH/dt = -0.1043 cells/mL/day
       dP/dt = 0.0643 cells/mL/day

    b. Figures





Unlike the carrying capacity model and the basic Lotka-Volterra model, this model both does not approach a steady-state where the predator and prey populations are fixed and does not remain at the starting oscillation/cycle amplitude. This is the most realistic model since it accounts for the timescale and efficiency with which predators consume prey. In the long-term, oscillations between both populations are expected since predators are not '100% efficient' at and do not 'immediately' consume prey. Thus, the model does not reach a steady state with fixed predator and prey populations (i.e. does not reach the nullcline intersection). However, this model still accounts for the environmental carrying capacity, seen as thresholds on the predator and prey population sizes. Depending on the initial populations, the number of predators and prey increase or decrease to approach an equilibrium oscillation/cycle, shown on the second figure.

CODE

```
close all
clc

%% Problem 1: Simulation of Lotka-Volterra model (DEMO)
disp('PROBLEM 1')
clear

%% Problem 1, Part A
% setting time, species values, and parameters
t = 0;
y = [0.4 0.4];
a = 0.5; b = 3; c = 3; m = 0.1;
p = [a b c m];

dydt = LV_model(t, y, p);
disp(dydt);

%% Problem 1, Part B
% setting timepoints, initial conditions, and parameter values
tspan = linspace(0, 500, 5000);
y0 = [0.4 0.4; 0.3 0.3; 0.2 0.2];

% setting up function for ode simulations
dydt = @(t, y) LV_model(t, y, p);

% initalizing storage for H and P
P = zeros(length(tspan), size(y0, 1));
H = P;

% setting tolerances for ode solver
options = odeset('AbsTol', 1e-8, 'RelTol', 1e-5);

% iterate through the initial conditions and simulate
for ii = 1:size(y0, 1)
    [~, y] = ode15s(dydt, tspan, y0(ii, :), options);
    P(:, ii) = y(:, 2);
    H(:, ii) = y(:, 1);
end

figure(1)
hold on
plot(tspan, H(:,1), LineWidth=1.5)
plot(tspan, P(:,2), LineWidth=1.5)
xlabel('Time')
ylabel('Species abundances')
title('PROBLEM 1B: H, P v. t')
legend('H', 'P', Location='best')
set(gca, 'Yscale', 'log')

figure(2)
```

```matlab
hold on
plot(H, P, LineWidth=1.5)
dHdt = @(H, P) a.*H - b.*H.*P;
dPdt = @(H, P) c.*H.*P - m.*P;
fimplicit(dHdt, '--k', LineWidth=1.5);
fimplicit(dPdt, '--k', LineWidth=1.5);
title('PROBLEM 1B: P v. H')
xlabel('H')
ylabel('P')
l = legend('(0.4, 0.4)', '(0.3, 0.3)', '(0.2, 0.2)', location='best');
title(l, '(H_0, P_0)')
set(gca, 'Yscale', 'log')
set(gca, 'Xscale', 'log')

%% Problem 1, Part C
% No MATLAB code required.


%% Problem 2: Introducing a carrying capacity on prey growth
disp('PROBLEM 2')
clear

%% Problem 2, Part A
t = 0;
y = [0.4 0.4];
a = 0.5; b = 3; c = 3; m = 0.1; K = 0.4;
p = [a b c m K];

dydt = carrying_capacity(t, y, p);
disp(dydt);

%% Problem 2, Part B
tspan = linspace(0, 500, 5000);
y0 = [0.4 0.4; 0.3 0.3; 0.2 0.2];

dydt = @(t, y) carrying_capacity(t, y, p);

P = zeros(length(tspan), size(y0, 1));
H = P;

options = odeset('AbsTol', 1e-8, 'RelTol', 1e-5);

for ii = 1:size(y0, 1)
    [~, y] = ode15s(dydt, tspan, y0(ii, :), options);
    P(:, ii) = y(:, 2);
    H(:, ii) = y(:, 1);
end

figure(3)
hold on
plot(tspan, H(:,1), LineWidth=1.5)
```

```
plot(tspan, P(:,2), LineWidth=1.5)
xlabel('Time')
ylabel('Species abundances')
title('PROBLEM 2B: H, P v. t')
legend('H', 'P', Location='best')
set(gca, 'Yscale', 'log')

figure(4)
hold on
plot(H, P, LineWidth=1.5)
dHdt = @(H, P) a.*H.*(1-(H./K)) - b.*H.*P;
dPdt = @(H, P) c.*H.*P - m.*P;
fimplicit(dHdt, '--k', LineWidth=1.5);
fimplicit(dPdt, '--k', LineWidth=1.5);
title('PROBLEM 2B: P v. H')
xlabel('H')
ylabel('P')
l = legend('(0.4, 0.4)', '(0.3, 0.3)', '(0.2, 0.2)', location='best');
title(l, '(H_0, P_0)')
set(gca, 'Yscale', 'log')
set(gca, 'Xscale', 'log')


%% Problem 3: Simulating a Macarthur-Rozenzweig model
disp('PROBLEM 2')
clear

%% Problem 3, Part A
t = 0;
y = [0.4 0.4];
a = 0.5; b = 3; eps = 1; tau = 3; m = 0.1; K = 0.4;
p = [a b eps tau m K];

dydt = MR_model(t, y, p);
disp(dydt);

%% Problem 3, Part B
tspan = linspace(0, 500, 5000);
y0 = [0.4 0.4; 0.3 0.3; 0.2 0.2];

dydt = @(t, y) MR_model(t, y, p);

P = zeros(length(tspan), size(y0, 1));
H = P;

options = odeset('AbsTol', 1e-8, 'RelTol', 1e-5);

for ii = 1:size(y0, 1)
    [~, y] = ode15s(dydt, tspan, y0(ii, :), options);
    P(:, ii) = y(:, 2);
    H(:, ii) = y(:, 1);
```

```
end

figure(5)
hold on
plot(tspan, H(:,1), LineWidth=1.5)
plot(tspan, P(:,2), LineWidth=1.5)
xlabel('Time')
ylabel('Species abundances')
title('PROBLEM 3B: H, P v. t')
legend('H', 'P', Location='best')
set(gca, 'Yscale', 'log')

figure(6)
hold on
plot(H, P, LineWidth=1.5)
dHdt = @(H, P) a.*H.*(1-(H./K)) - (b.*H.*P)./(1+(b.*tau.*H));
dPdt = @(H, P) (eps.*b.*H.*P)./(1+(b.*tau.*H)) - m.*P;
fimplicit(dHdt, '--k', LineWidth=1.5);
fimplicit(dPdt, '--k', LineWidth=1.5);
title('PROBLEM 3B: P v. H')
xlabel('H')
ylabel('P')
l = legend('(0.4, 0.4)', '(0.3, 0.3)', '(0.2, 0.2)', location='best');
title(l, '(H_0, P_0)')
set(gca, 'Yscale', 'log')
set(gca, 'Xscale', 'log')


%% Functions

% Problem 1
function dydt = LV_model(t, y, p)
% Rate laws for standard LV model
    % setting species
    H = y(1);
    P = y(2);

    % setting parameters
    a = p(1); b = p(2); c = p(3); m = p(4);

    % calculating rates of change
    dydt = zeros(size(y));

    % dHdt
    dydt(1) = a*H - b*H*P;

    % dPdt
    dydt(2) = c*H*P - m*P;
end

% Problem 2
```

```
function dydt = carrying_capacity(t, y, p)
    H = y(1);
    P = y(2);

    a = p(1); b = p(2); c = p(3); m = p(4); K = p(5);

    dydt = zeros(size(y));
    dydt(1) = a*H*(1-(H/K)) - b*H*P;
    dydt(2) = c*H*P - m*P;
end

% Problem 3
function dydt = MR_model(t, y, p)
    H = y(1);
    P = y(2);

    a = p(1); b = p(2); eps = p(3); tau = p(4); m = p(5); K = p(6);

    dydt = zeros(size(y));
    dydt(1) = a*H*(1-(H/K)) - (b*H*P)/(1+(b*tau*H));
    dydt(2) = (eps*b*H*P)/(1+(b*tau*H)) - m*P;
end
```