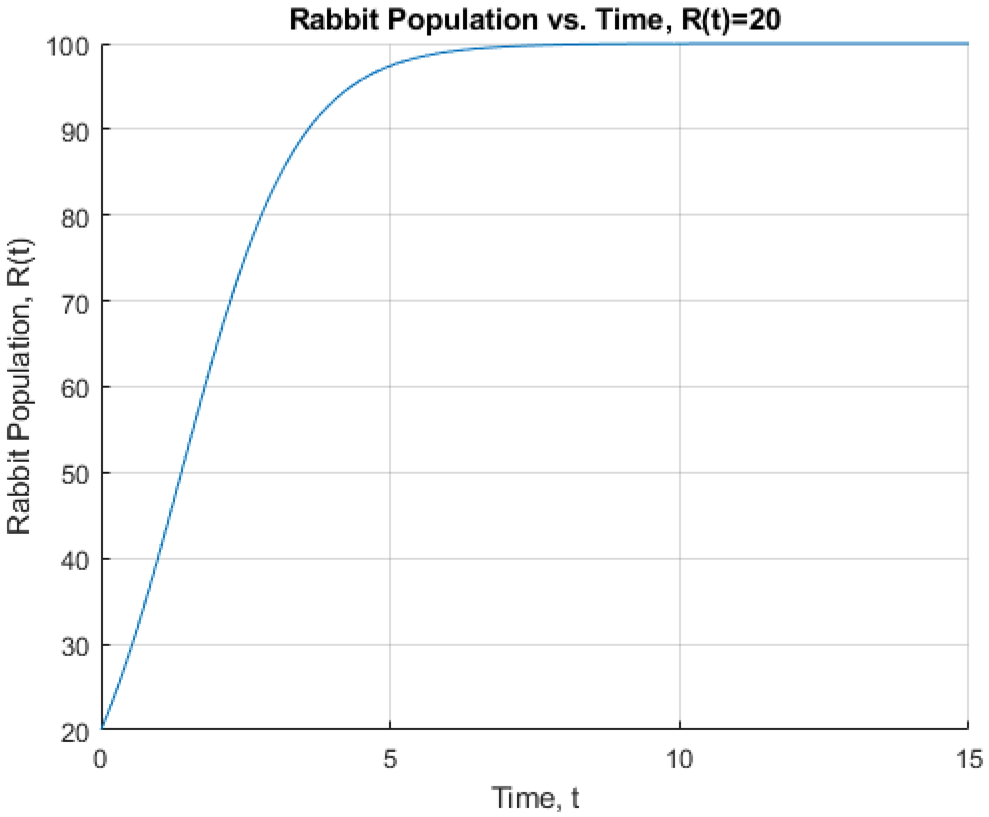
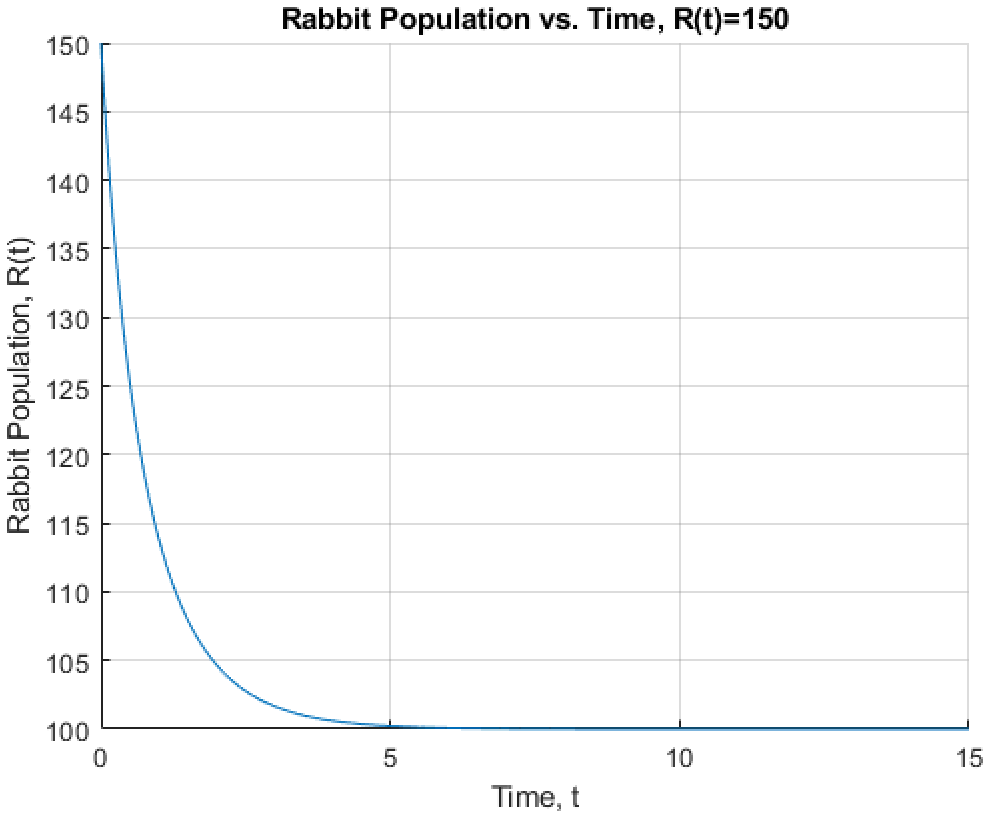
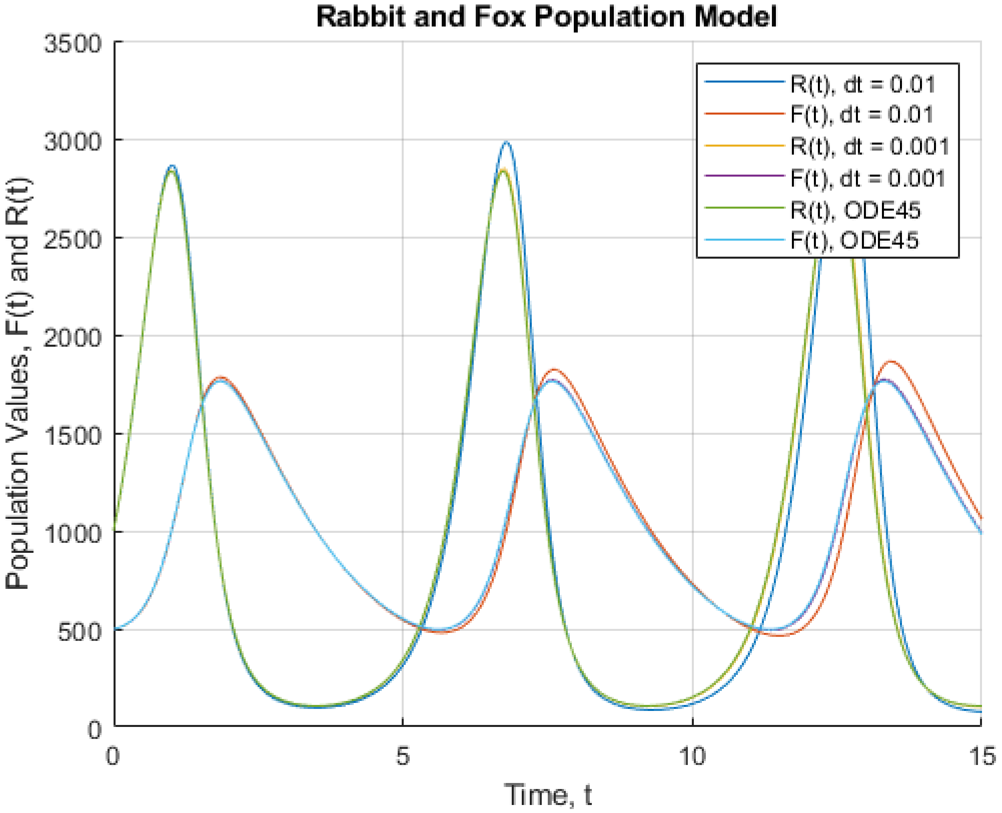
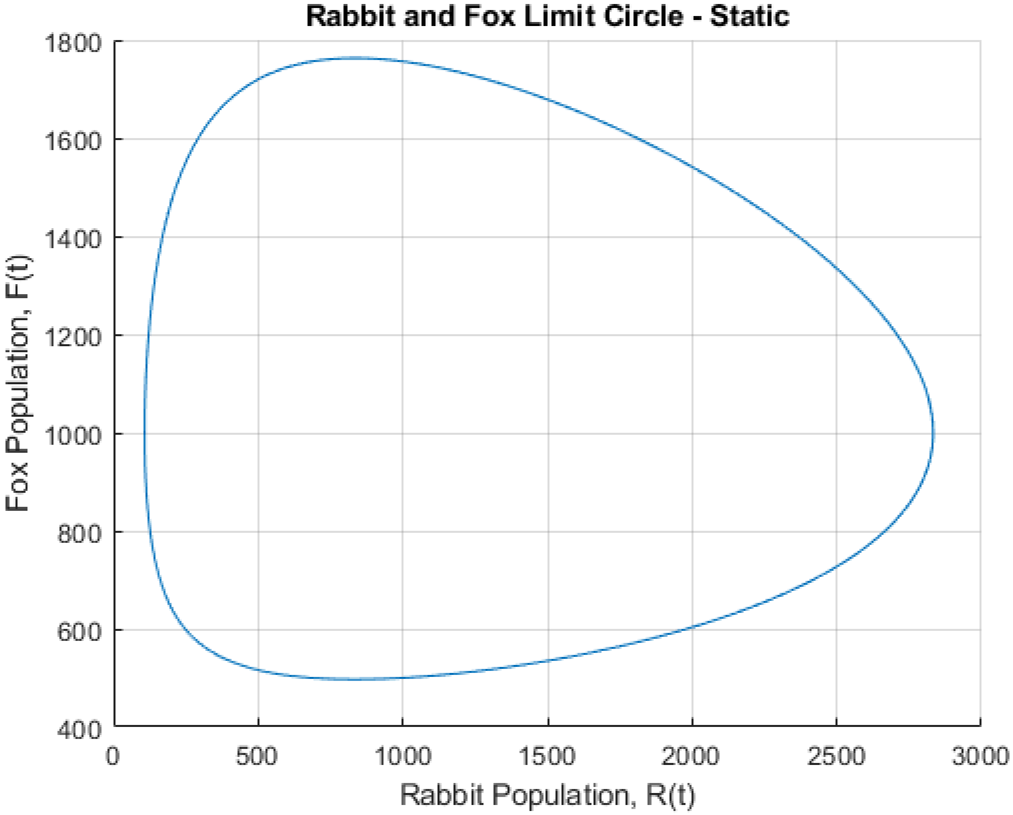
```
% Pierce Zhang, CMOR220, Fall 2023, Project 5: Population Model
% population_models.m
% Solve ODEs and model population of rabbits and foxes in two ways
% Last modified: 11 October 2023

% Driver function
function population_models
    figure(); hold on; grid on;
    % r = 1, R(0) = 20, x in [0, 15], delta = 0.01
    [t_Ra, Ra] = eulers_method(@R_prime, 0, 20, 15, 0.01);
    plot(t_Ra, Ra);
    xlabel("Time, t"); ylabel("Rabbit Population, R(t)"); title("Rabbit
 Population vs. Time, R(t)=20");
    % R(0) = 150
    figure(); hold on; grid on;
    [t_Rb, Rb] = eulers_method(@R_prime, 0, 150, 15, 0.01);
    plot(t_Rb, Rb);
    xlabel("Time, t"); ylabel("Rabbit Population, R(t)"); title("Rabbit
 Population vs. Time, R(t)=150");

    % (Rb) QUESTION ANSWERED: The rabbit population will tend towards K = 100
    % over time.

    % R(0) = 1000; F(0) = 500
    [t_sys_1, R_sys_1, F_sys_1] = eulers_method_sys(@R_sys_prime,
 @F_sys_prime, 1000, 500, 0, 15, 0.01);
    [t_sys_2, R_sys_2, F_sys_2] = eulers_method_sys(@R_sys_prime,
 @F_sys_prime, 1000, 500, 0, 15, 0.001);

    options = odeset('RelTol', 1e-6);
    [ts, RFs] = ode45(@odefun, [0 15], [1000 500], options);

    figure(); hold on; grid on;
    plot(t_sys_1, R_sys_1); plot(t_sys_1, F_sys_1)
    plot(t_sys_2, R_sys_2); plot(t_sys_2, F_sys_2);
    plot(ts, RFs);
    xlabel("Time, t"); ylabel("Population Values, F(t) and R(t)");
    title("Rabbit and Fox Population Model");
    legend("R(t), dt = 0.01","F(t), dt = 0.01","R(t), dt = 0.001", ...
        "F(t), dt = 0.001","R(t), ODE45","F(t), ODE45");

    % (RFc) QUESTION ANSWERED: There are three categories of trendlines in
    % this plot: the ODE solutions by Euler's method at dt = 0.01, dt =
    % 0.001, and using ode45. The ODE solutions from Euler's method with dt
    % = 0.01 had noticeably more deviation from the solutions produced by
    % dt = 0.001, which in turn had more deviation from the solutions
    % produced by ode45. This is likely due to the increase in accuracy
    % afforded to smaller t intervals which capture more nuanced behavior
    % and reduce error from linear skipping. In general, we see that the
    % population of both animals are mostly periodic over time.

    figure(); hold on; grid on;
    plot(RFs(:,1), RFs(:,2));
    xlabel("Rabbit Population, R(t)"); ylabel("Fox Population, F(t)");
    title("Rabbit and Fox Limit Circle - Static");
    % (RFd) QUESTION ANSWERED: This plot shows a loop which is the
    % population limit for F(t) given R(t) at any given time, or vice versa
    % depending on how you want to interpret it. Basically, this plot
    % contains the same data as in RFc but F(t) and R(t) are plotted
    % against each other instead of being plotted as dependent variables
    % against time.

    figure();
    LimitCircleVid = VideoWriter("LimitCircle.avi");
    open(LimitCircleVid);
    for n = 1:length(ts)
        hold on; grid on;
        xlabel("Rabbits"); ylabel("Foxes");
        title("time: " + num2str(n/length(ts) * 15));
        plot(RFs(:,1), RFs(:,2));
        plot(RFs(n,1), RFs(n,2),"o","MarkerEdgeColor","r");
        writeVideo(LimitCircleVid, getframe(gcf));
        clf;
     end
     close(LimitCircleVid);
end

% Inputs: x, y, values at which to evaluate the first R' function
% Outputs: value of R'(x, y)
function [value] = R_prime(x, y)
    r = 1; K = 100;
    value = r * y * (1 - y/K);
end

% R is x, F is y
% Inputs: x, y, values at which to evaluate the systemic R' function
% Outputs: value of systemic R'(x, y)
function [value] = R_sys_prime(x, y)
    k1 = 3; k2 = 3e-3;
    value = k1 * x - k2 * x * y;
end

% Inputs: x, y, values at which to evaluate the systemic F' function
% Outputs: value of systemic F'(x, y)
function [value] = F_sys_prime(x, y)
    k3 = 6e-4; k4 = 0.5;
    value = k3 * x * y - k4 * y;
end

% Inputs: x, y, values at which to evaluate the ODE version of RF' function
% Outputs: vector containins values of systemic RF'(x, y)
function dydt = odefun(t,y)
    dydt = zeros(2,1);
    dydt(1) = R_sys_prime(y(1), y(2));
    dydt(2) = F_sys_prime(y(1), y(2));
end
```

```
function [xs, ys] = eulers_method(fun, x0, y0, xf, dx)
    % This function uses Euler's method to approximate the solution to an ODE.
    % Inputs:
    % - fun: The function representing the ODE.
    % - x0: Initial value of x.
    % - y0: Initial value of y.
    % - b: The upper limit of the x-range for the approximation.
    % - dx: The step size.
    % Output:
    % - y: The approximate solution to the ODE.
    xs = x0:dx:xf;
    ys = zeros(1, length(xs));
    ys(1) = y0;
    for n = 1:length(xs) - 1
        dy = (fun(xs(n), ys(n))) * dx;
        ys(n + 1) = ys(n) + dy;
    end
end

function [ts, xs, ys] = eulers_method_sys(funx, funy, x0, y0, t, tf, dt)
    % This function uses Euler's method to approximate a system of ODEs.
    % Inputs:
    % - funx: The function representing the first ODE.
    % - funy: The function representing the second ODE.
    % - x0: Initial value of x.
    % - y0: Initial value of y.
    % - t: Initial time.
    % - tf: Final time.
    % - dt: The time step.
    % Output:
    % - x: The approximate solution for x.
    % - y: The approximate solution for y.
    ts = t:dt:tf;
    xs = zeros(1, length(ts));
    ys = zeros(1, length(ts));
    xs(1) = x0;
    ys(1) = y0;
    for n = 1:length(ts) - 1
        dx = (funx(xs(n), ys(n))) * dt;
        dy = (funy(xs(n), ys(n))) * dt;
        xs(n + 1) = xs(n) + dx;
        ys(n + 1) = ys(n) + dy;
    end
end
```


Rabbit Population vs. Time, R(t)=20
Rabbit Population, R(t)
Time, t
100
90
80
70
60
50
40
30
20
0
5
10
15


Rabbit Population vs. Time, R(t)=150
Rabbit Population, R(t)
Time, t
150
145
140
135
130
125
120
115
110
105
100
0
5
10
15


Rabbit and Fox Population Model
Population Values, F(t) and R(t)
Time, t
0
500
1000
1500
2000
2500
3000
3500
0
5
10
15
R(t), dt = 0.01
F(t), dt = 0.01
R(t), dt = 0.001
F(t), dt = 0.001
R(t), ODE45
F(t), ODE45


Rabbit and Fox Limit Circle - Static
Fox Population, F(t)
Rabbit Population, R(t)
400
600
800
1000
1200
1400
1600
1800
0
500
1000
1500
2000
2500
3000

Published with MATLAB® R2023a