Table of Contents

```
%%%% Differential Equations Homework 4 - Romeo Perlstein %%%%

close all
close all force
```

Problem F9

a)

```
syms x y % set up some global sim variables
eq1 = x * (1 - 0.5*x - 0.5*y); % this be equation 1 (yarr)
eq2 = y * (-0.25 + 0.5*x); % this is equation 2
[xc,yc] = solve(eq1, eq2, [x,y]), % solve for da equation symbolically

critpoints = [xc yc] % these are the critical points

JacobianMatrix = jacobian([eq1 eq2], [x y]) % so excited to be using
 Jacobians!
eigens = eig(JacobianMatrix);
eigens_solved1 = subs(eigens, {x,y}, {0,0}) % plugging in crit points to eigen
 val eq
eigens_solved2 = subs(eigens, {x,y}, {2,0})
eigens_solved3 = subs(eigens, {x,y}, {1/2, 3/2})

disp("eigen values for [0,0] crit point") % displaying for the graders liesure
 (Hey Revati?)
disp(eigens_solved1)
disp("eigen values for [2,0] crit point")
disp(eigens_solved2)
disp("eigen values for [1/2,3/2] crit point")
disp(eigens_solved3)

% for [0,0], since the eigenvalues are opposite in sign, the critical point
% is a saddle
% for [2,0], since again since the eigenvalues are opposite in sign, the
% critical point is a saddle
% for [1/2, 3/2], since the critcal point is a complex number, that has the
% form a +/- bi, and since a is less than 0 in both cases, the critical
% point is a stable spiral
```

b)

```
[X, Y] = meshgrid(-2:1/4:3, -2:1/4:3); % makin da graph
U = X.*(1 - 0.5*X - 0.5*Y);
V = Y.*(-0.25 + 0.5*X);
L = sqrt((U/3).^2 + (V/3).^2);
vectorField = quiver(X, Y, U./L, V./L, .5, "*bl") % blue vector field
axis equal tight % tite B)



figure
vectorField_with_points = quiver(X, Y, U./L, V./L, .5, "b") % plot again but
 this time with labeled points
hold on
plot(xc(1), yc(1), "or")
plot(xc(2), yc(2), "or")
plot(xc(3), yc(3), "or")
axis equal tight
```

c)

```
f = @(t,x) [x(1)*(1-.5*x(1)-.5*x(2)); x(2)*(-.25 + .5*x(1))]; % no find the
 phase portrait (curtosey of the textbook)
warning off MATLAB:ode45:IntegrationTolNotMet
for a = [1 5]
    for b = 0.1:0.015:1
        [t, xa] = ode45(f, [0,50], [a*b a*4*(.5-b)]);
        plot(xa(:,1), xa(:,2))
        [t, xa] = ode45(f, [0 -10], [a*b a*4*(.5-b)]);
        plot(xa(:,1), xa(:,2))
    end
end
axis([-2 3 -2 3])
% flow of the field above 0 seems to be going toward the critical point,
% spiraling inwards, from right to left!
% as well from below x=0, and y>0, the graph starts to trend toward the
% bottom left corner, for seemingly infinity
% it reminds me a lot of hair, or pasta!
```

d)

My interpretation of the phase portrait is the following: as as predator population (y) approaches 0 in the first quadrant, the prey population begins to increase, and vice versa for their respective axis, however if the prey population is equal to 0, the predator population begins to decrease. Invertedly, if the predator population decrease, the prey population increases. However, if the predator population is equal to 0, the prey population does not increase exponentially, and starts to decrease after the prey population begins to grow (I'm guessing do to lack of resources. Also, the prey population will decrease as long as the predators are greater than 0, and it's a more rapid increase if there are more prey, but proportionally the decrease of large amounts of prey is proportional. What seems to happen, is that the population sizes, no matter the start point (as long as it's in the first quadrant), will always approach an equilibrium point after a certain time. where the rate of change of predator size is "constant" with the rate of change of the prey size. Conceptually, this all makes sense: if the predator size is large, but the prey size is small, then the predator population will decrease, and if the prey size can only be so large before competition of resources causes them to also decrease. And then of course, there would be an equilibrium point where the change in predator size is constsistant and the change in prey size is consistant as well. This is what I was able to take away from the phase portrait

e)

```
figure
[t, xb] = ode45(f, [0:.1:5], [1,1]) % begin solving for the initial given
 value

[X1, Y1] = meshgrid(0:1/4:3, 0:1/4:3); % (be lazy) remake the og graph so you
 can "hold"
U = X1.*(1 - 0.5*X1 - 0.5*Y1);
V = Y1.*(-0.25 + 0.5*X1);
L = sqrt((U/3).^2 + (V/3).^2);
quiver(X1, Y1, U./L, V./L, .5, "r")
axis tight
hold on
plot(xb(:,1), xb(:,2), "b")

[t, xc] = ode45(f, [0:1:5], [1,1]) % solve da equation again to soley extract
 1 2 3 4 5
table_of_vals = [t(1) xc(1,1) xc(1,2) ; t(2) xc(2,1) xc(2,2); t(3) xc(3,1)
 xc(3,2); t(4) xc(4,1) xc(4,2); t(5) xc(5,1) xc(5,2); t(6) xc(6,1) xc(6,2)] %
 do a little extractoin
```

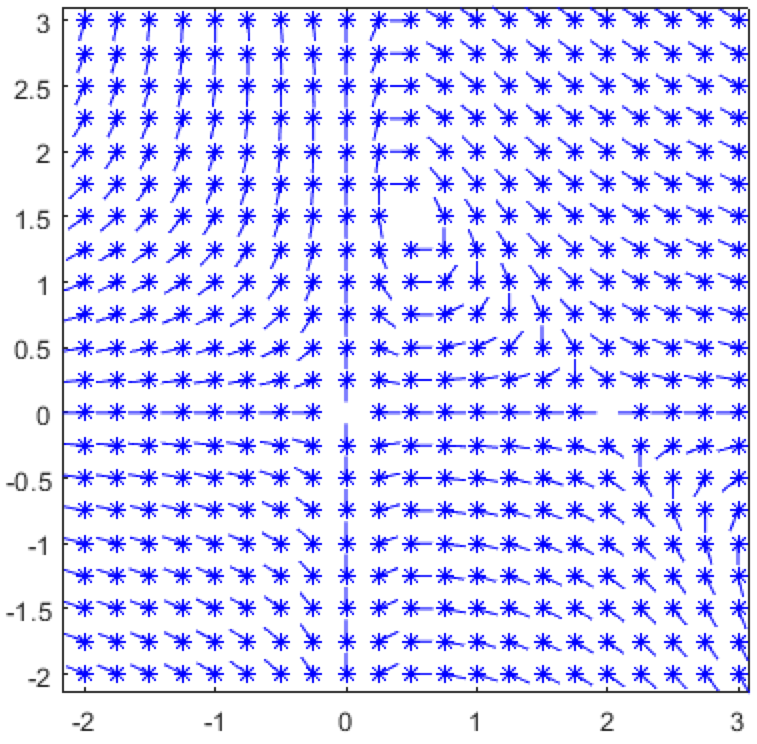
f)

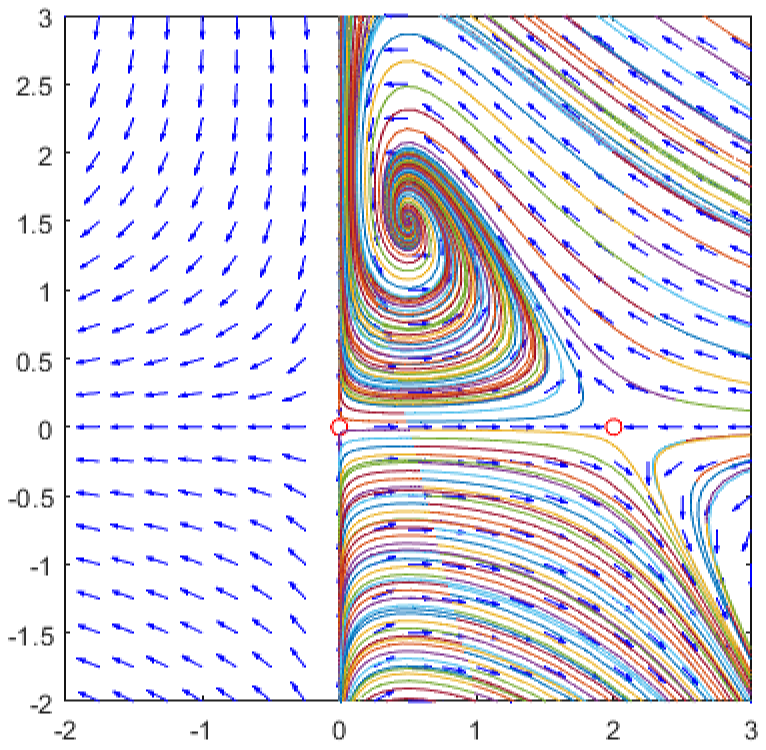
making the roughest estimation known to man, and using the table creating from solving the IVP, I think it would take roughly 3-4 timesteps before each population is within .1 of their equilibrium points. This, again is a rough estimate going off of the table date.

```
% hey revati! I hope things are well. Sorry for not being in discussion,
% like, at all. I've been putting this class on the back burner more then I
% shuold have, but wish me luck! -Romeo
```

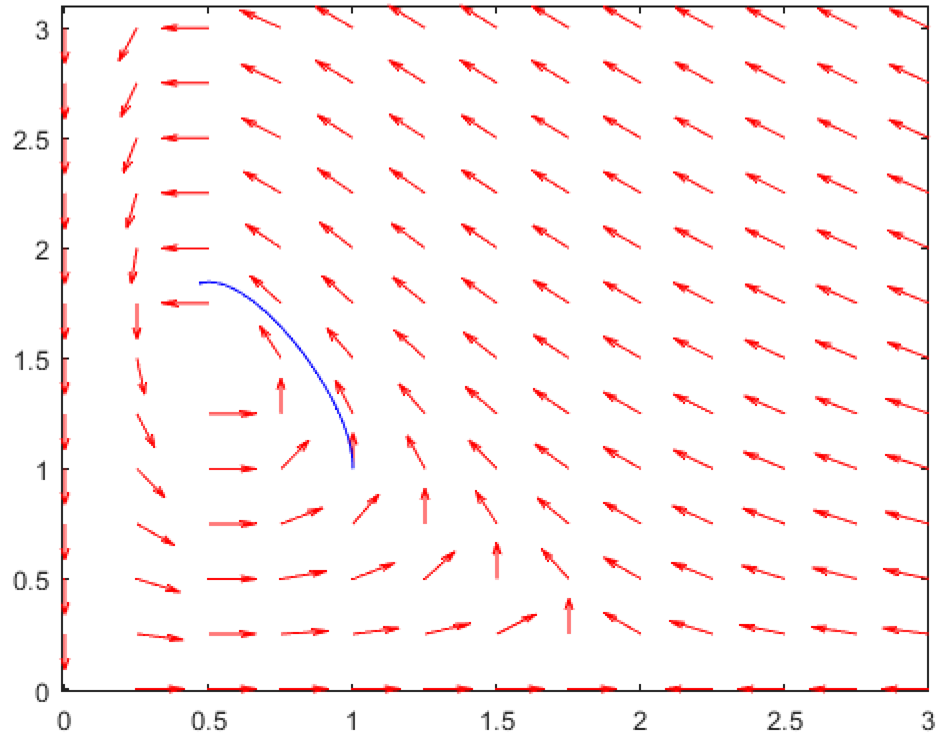
```
xc =

  0
  2
1/2


yc =

  0
  0
```

```
3/2


critpoints =

[  0,   0]
[  2,   0]
[1/2, 3/2]


JacobianMatrix =

[1 - y/2 - x,      -x/2]
[        y/2, x/2 - 1/4]


eigens_solved1 =

-1/4
   1


eigens_solved2 =

 -1
3/4


eigens_solved3 =

- (11^(1/2)*1i)/8 - 1/8
  (11^(1/2)*1i)/8 - 1/8

eigen values for [0,0] crit point
-1/4
   1

eigen values for [2,0] crit point
 -1
3/4

eigen values for [1/2,3/2] crit point
- (11^(1/2)*1i)/8 - 1/8
  (11^(1/2)*1i)/8 - 1/8


vectorField =

  Quiver with properties:

        Color: [0 0 1]
    LineStyle: '-'
    LineWidth: 0.5000
        XData: [21×21 double]
        YData: [21×21 double]
        ZData: []
        UData: [21×21 double]
        VData: [21×21 double]
        WData: []

  Use GET to show all properties


vectorField_with_points = 

  Quiver with properties:

        Color: [0 0 1]
    LineStyle: '-'
    LineWidth: 0.5000
        XData: [21×21 double]
        YData: [21×21 double]
        ZData: []
        UData: [21×21 double]
        VData: [21×21 double]
        WData: []

  Use GET to show all properties


t = 

         0
    0.1000
    0.2000
    0.3000
    0.4000
    0.5000
    0.6000
    0.7000
    0.8000
    0.9000
    1.0000
    1.1000
    1.2000
    1.3000
    1.4000
    1.5000
    1.6000
    1.7000
    1.8000
    1.9000
    2.0000
    2.1000
    2.2000
    2.3000
    2.4000
    2.5000
    2.6000
    2.7000
    2.8000
    2.9000
    3.0000
    3.1000
    3.2000
    3.3000
    3.4000
    3.5000
    3.6000
    3.7000
    3.8000
    3.9000
    4.0000
    4.1000
    4.2000
    4.3000
    4.4000
    4.5000
    4.6000
    4.7000
    4.8000
    4.9000
    5.0000


xb =

    1.0000    1.0000
    0.9994    1.0253
    0.9975    1.0512
    0.9945    1.0776
    0.9904    1.1045
    0.9852    1.1317
    0.9789    1.1593
    0.9716    1.1872
    0.9635    1.2153
    0.9544    1.2435
    0.9446    1.2718
    0.9340    1.3000
    0.9228    1.3282
    0.9109    1.3562
    0.8985    1.3839
    0.8855    1.4113
    0.8722    1.4383
    0.8585    1.4648
    0.8445    1.4908
    0.8302    1.5161
    0.8157    1.5408
    0.8011    1.5648
    0.7864    1.5879
    0.7717    1.6102
    0.7570    1.6316
    0.7423    1.6521
    0.7277    1.6717
    0.7132    1.6902
    0.6989    1.7077
    0.6848    1.7242
    0.6709    1.7396
    0.6573    1.7539
    0.6439    1.7671
    0.6308    1.7793
    0.6181    1.7904
    0.6056    1.8005
    0.5935    1.8094
    0.5817    1.8174
    0.5703    1.8243
    0.5592    1.8302
    0.5485    1.8351
    0.5381    1.8391
    0.5282    1.8422
    0.5186    1.8443
    0.5094    1.8456
    0.5005    1.8461
    0.4920    1.8457
    0.4839    1.8446
    0.4761    1.8427
    0.4687    1.8402
    0.4616    1.8370


t =

     0
     1
     2
     3
     4
     5


xc =

    1.0000    1.0000
    0.9446    1.2718
    0.8157    1.5408
    0.6709    1.7396
    0.5485    1.8351
    0.4616    1.8370


table_of_vals =

         0    1.0000    1.0000
    1.0000    0.9446    1.2718
    2.0000    0.8157    1.5408
    3.0000    0.6709    1.7396
    4.0000      0.5485      1.8351
    5.0000      0.4616      1.8370
```





3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
3



3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3

Published with MATLAB® R2022b