(a) A simple Lotka-Volterra Model

Assume the Lotka-Volterra Model:

$$\frac{dN_{prey}}{dt} = R_{o,prey} \cdot N_{prey}(t) - \gamma \cdot N_{prey}(t) \cdot N_{pred}(t)$$

$$\frac{dN_{pred}}{dt} = \epsilon \cdot \gamma \cdot N_{prey}(t) \cdot N_{pred}(t) - R_{o,pred} \cdot N_{pred}(t)$$

Prey are rabbits and predator are foxes.

Consider the parameters as follow:

$$R_{o,prey} = 0.04$$

$$R_{o,pred} = 0.2$$

$$\gamma = 0.0005$$

$$\epsilon = 0.1$$

Assume also that the time units are all in days and that the populations are number f individuals per square kilometer.

In case of no foxes, rabbits would being breeded by exponent as shown below:

$$\frac{\boldsymbol{dN_{prey}}}{\boldsymbol{dt}} = \boldsymbol{R_{o,prey}} \cdot \boldsymbol{N_{prey}}(\boldsymbol{t}) \rightarrow \frac{dN_{prey}}{N_{prey}} = R_{o,prey} \cdot dt \rightarrow \ln N_{prey}(t) - \ln N_{prey}(t=0) = R_{o,prey} \cdot t$$

$$\rightarrow \ln \frac{N_{prey}(t)}{N_{prey}(t=0)} = R_{o,prey} \cdot t \rightarrow \boldsymbol{N_{prey}}(\boldsymbol{t}) = \boldsymbol{N_{prey}}(\boldsymbol{t=0}) \cdot \boldsymbol{e^{R_{o,prey} \cdot t}}$$

Rabbits doubling time is may be easily found from the solution given above:

$$\ln \frac{N_{prey}(t)}{N_{prey}(t=0)} = R_{o,prey} \cdot t$$

$$N_{prey}(t_2) = 2 \cdot N_{prey}(t=0) \rightarrow R_{o,prey} \cdot t_2 = \ln 2 \rightarrow \boldsymbol{t_2} = \frac{\ln 2}{R_{o,prey}} = \frac{\ln 2}{0.04} \approx \boldsymbol{17.33\ days}$$

In case of no rabbits, foxes will die same by exponent:

$$\frac{\boldsymbol{dN_{pred}}}{\boldsymbol{dt}} = -\boldsymbol{R_{o,pred}} \cdot \boldsymbol{N_{pred}}(\boldsymbol{t}) \rightarrow \frac{dN_{pred}}{N_{pred}} = -R_{o,pred} \cdot dt \rightarrow \ln N_{pred}(t) - \ln N_{pred}(t=0)$$

$$= -R_{o,pred} \cdot t \rightarrow \ln \frac{N_{pred}(t)}{N_{pred}(t=0)} = -R_{o,pred} \cdot t \rightarrow \boldsymbol{N_{pred}}(\boldsymbol{t}) = \boldsymbol{N_{pred}}(\boldsymbol{t} = \boldsymbol{0}) \cdot \boldsymbol{e^{-R_{o,pred} \cdot t}}$$

Foxes halving rate may be easily found from the solution given above:

$$\ln \frac{N_{pred}(t)}{N_{pred}(t=0)} = -R_{o,pred} \cdot t$$

$$N_{pred}\left(t_{1/2}\right) = \frac{1}{2} \cdot N_{pred}(t=0) \rightarrow R_{o,pred} \cdot t_{\frac{1}{2}} = \ln 2 \rightarrow \boldsymbol{t_{\frac{1}{2}}} = \frac{\ln 2}{R_{o,pred}} = \frac{\ln 2}{0.2} \approx \boldsymbol{3.47\ days}$$

Predators can't leave without prey – they need to eat something: without prey predators die. The prey can't being breeded infinitely – they also need food and this food is enough to feed only some critical amount of prey, but it is not counted in this model.

Parameters $R_{o,prey}$ and $R_{o,pred}$ are rates of rabbits birth and foxes death respectively and they, at least, seems OK in sense of order but not accurate – **17 days to double the population is less then expected (rabbits pregnancy lasts for about a month – and after birth those rabbits should growth some time to became fertile)**

Other parameters define the interactions between prey and predators- $\boldsymbol{\gamma}$ is some kind of successful hunting probability coefficient (it defines how strongly predators presence influences on prey), $\boldsymbol{\epsilon}$ is some kind of hunted rabbits transformation factor (it defines how strongly hunted prey are transforming into new predators). The given values is also are OK by order but a bit strange and seems not accurate.

Balance or static equilibrium in the system means zero derivative:

$$\frac{dN_{prey}}{dt} = 0 = R_{o,prey} \cdot N_{prey} - \gamma \cdot N_{prey} \cdot N_{pred}$$

$$\frac{dN_{pred}}{dt} = 0 = \epsilon \cdot \gamma \cdot N_{prey} \cdot N_{pred} - R_{o,pred} \cdot N_{pred}$$

That corresponds to:

$$N_{pred} = \frac{R_{o,prey}}{\gamma} = \frac{0.04}{0.0005} = 80$$

$$N_{prey} = \frac{R_{o,pred}}{\epsilon \cdot \gamma} = \frac{0.2}{0.1 \cdot 0.0005} = 4000$$

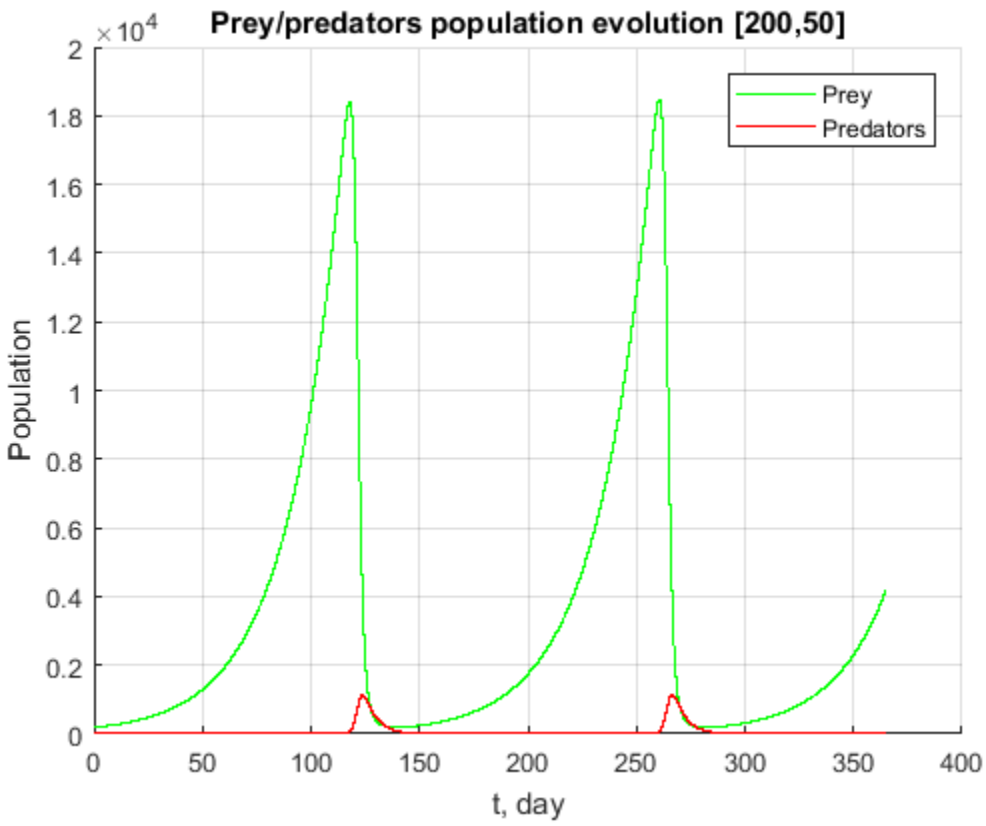
To investigate the given model it was proposed to use numerical approaches (Euler forward method) with three initial conditions:

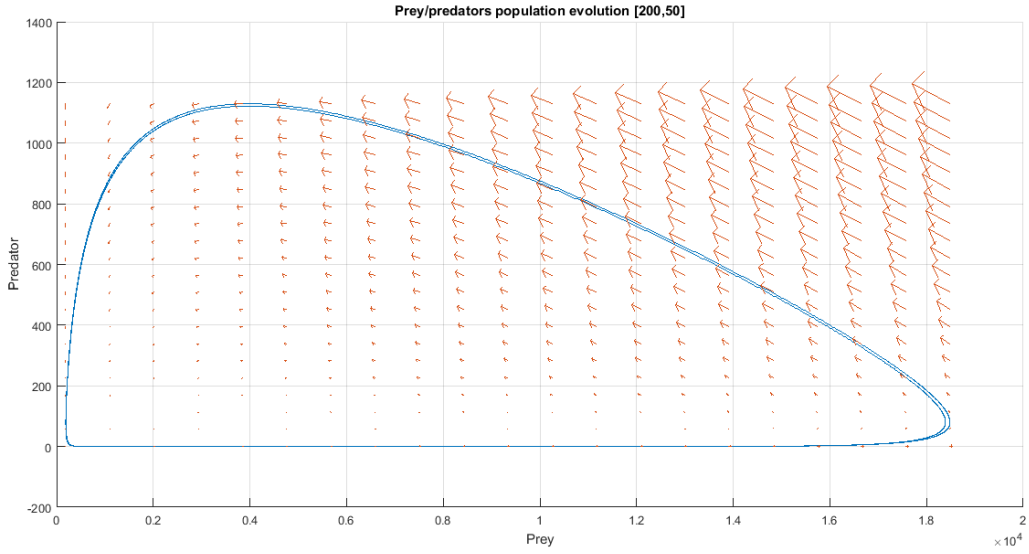
$$N_{pred}(t=0) = \{200, \quad 5000, \quad 4000\}$$

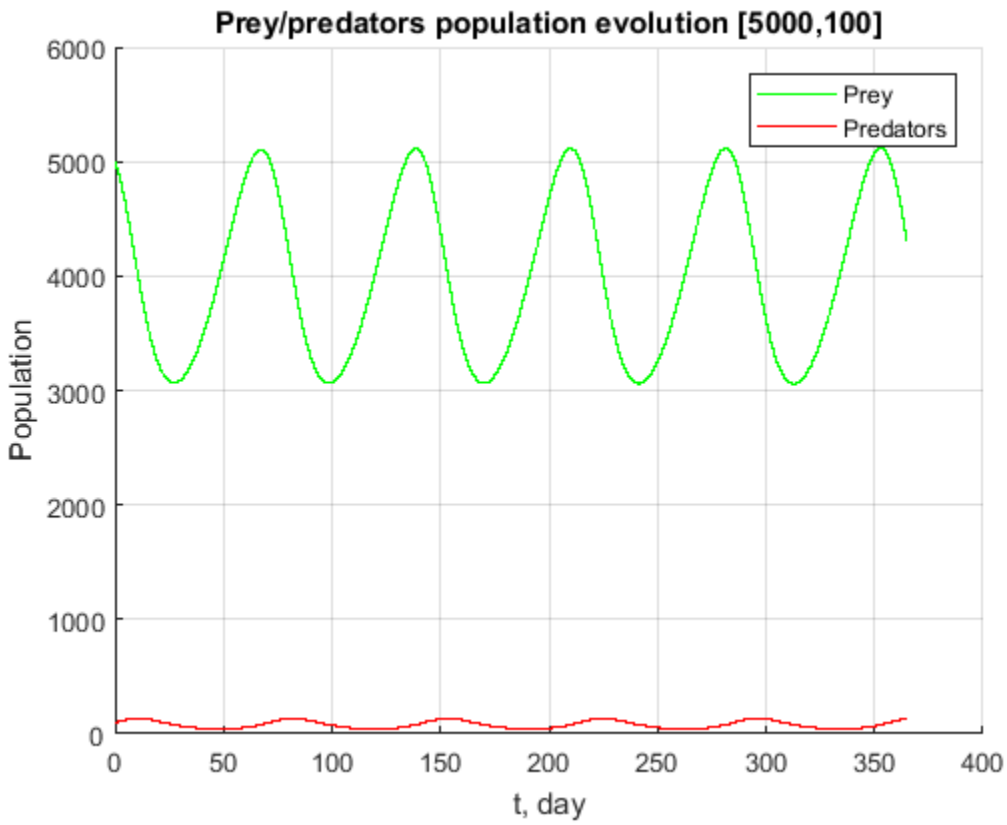
$$N_{prey}(t=0) = \{50, \quad 100, \quad 80\}$$

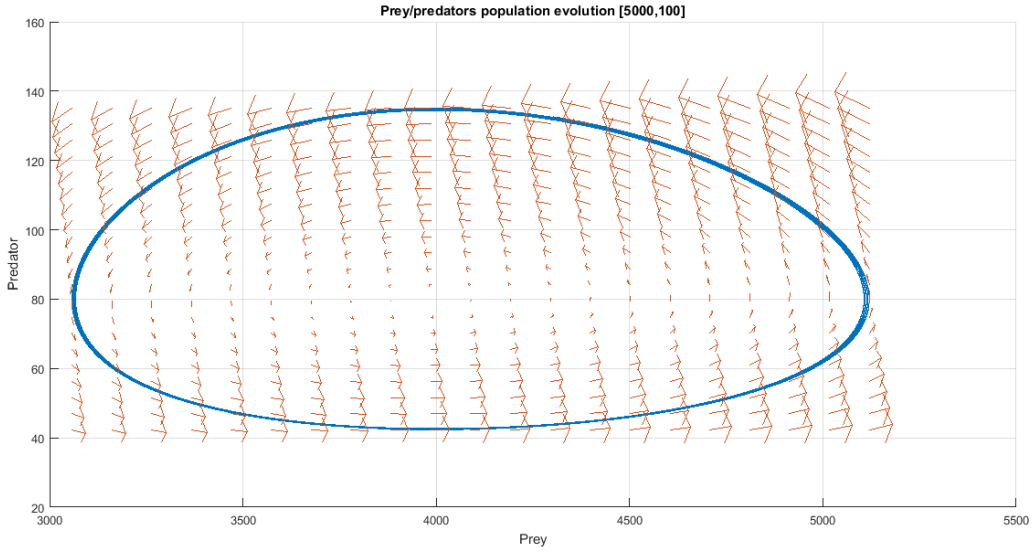
The idea is implemented as MATLAB script (code is attached in the appendix section – Appendix A).

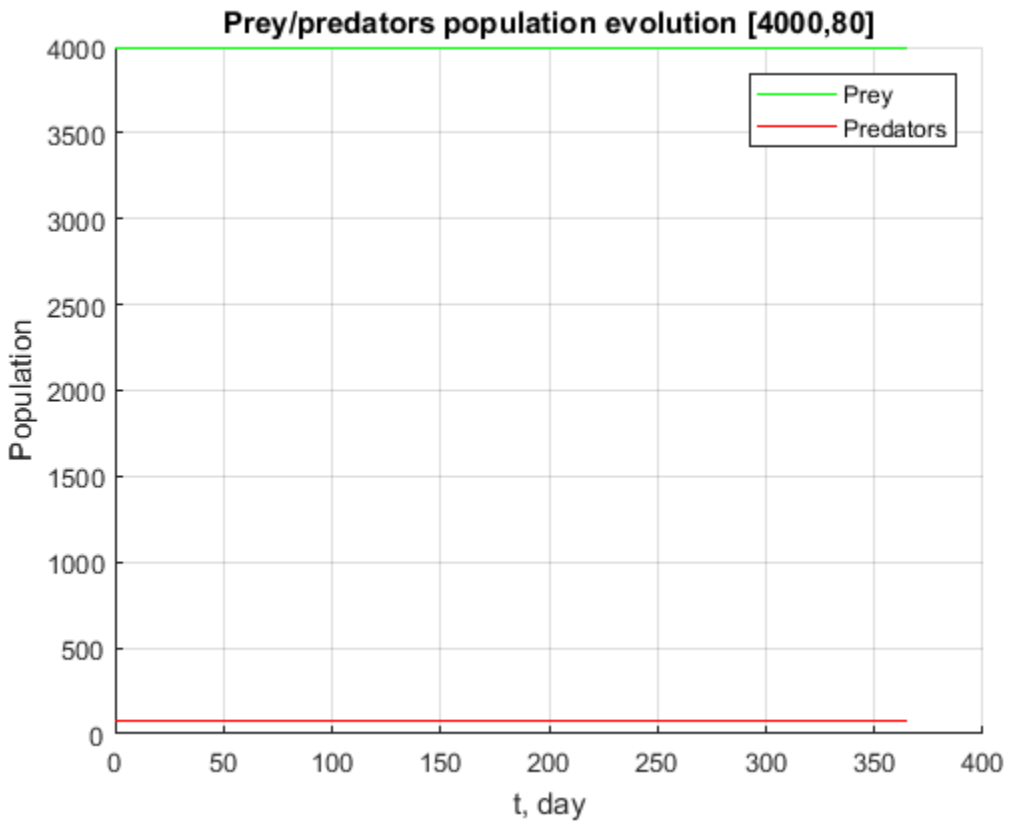
The results of the simulation are shown below:

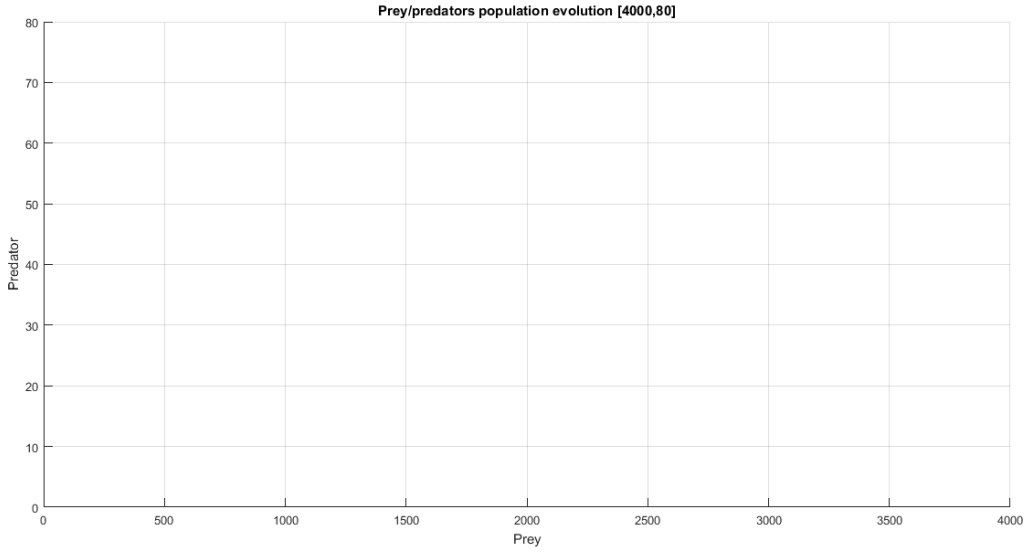





Prey/predators population evolution [5000,100]
Prey
Predators
Population
t, day
0
1000
2000
3000
4000
5000
6000
0
50
100
150
200
250
300
350
400


Prey/predators population evolution [5000,100]
Predator
Prey
20
40
60
80
100
120
140
160
3000
3500
4000
4500
5000
5500





As it is seen from figures above three different initial conditions have led to different situations:

- $\{200, 50\}$ correspond to inharmonic situation - prosperity alternates with extinction
- $\{5000, 100\}$ correspond to harmonic situation (dynamic equilibrium) – classical situation, population of both prey and predators are slightly oscillate around stable points.

- {4000,80}corresponds to equilibrium – just as expected (in this point the derivatives equal to zero)

(b) Extending the Lotka-Volterra Model

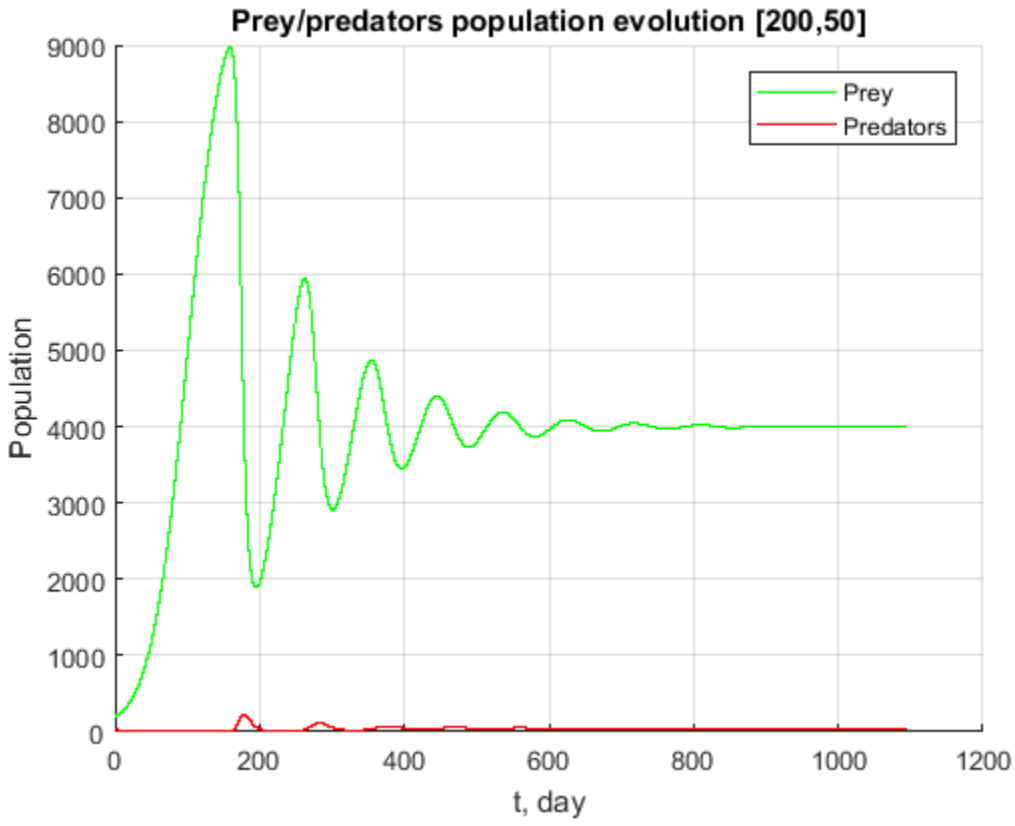
To make described model more realistic it was proposed to update it as follows:

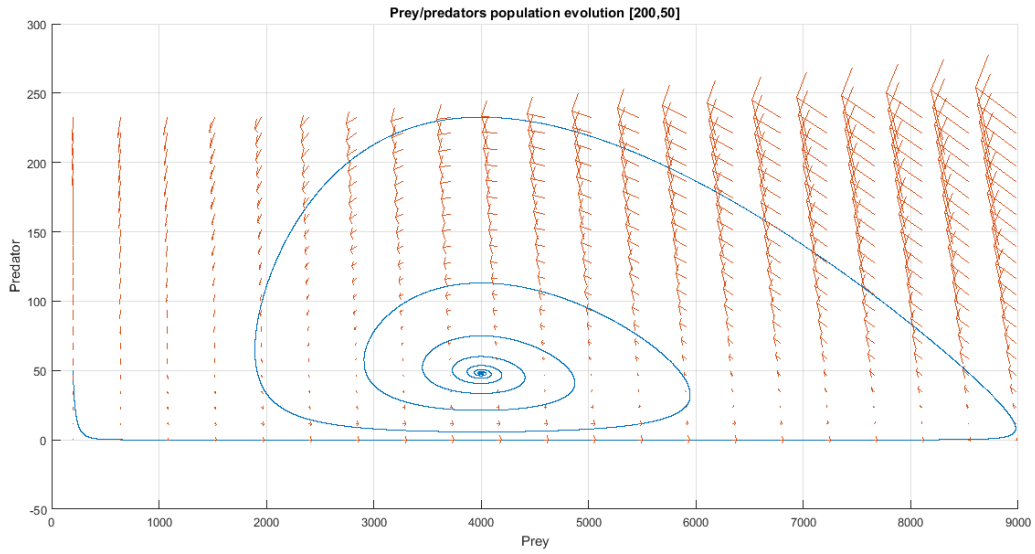
$$\frac{dN_{prey}}{dt} = R_{o,prey} \cdot (\mathbf{1} - \frac{\boldsymbol{N_{prey}(t)}}{\boldsymbol{K}}) \cdot N_{prey}(t) - \gamma \cdot N_{prey}(t) \cdot N_{pred}(t)$$

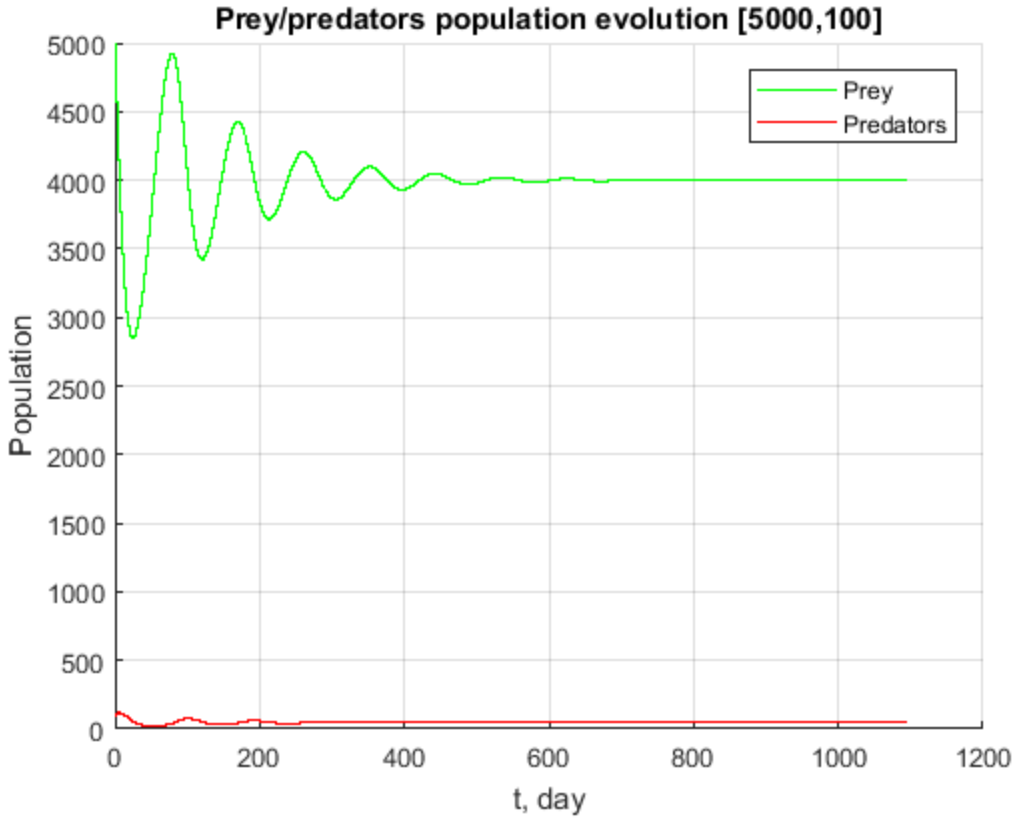
$$\frac{dN_{pred}}{dt} = \epsilon \cdot \gamma \cdot N_{prey}(t) \cdot N_{pred}(t) - R_{o,pred} \cdot N_{pred}(t)$$

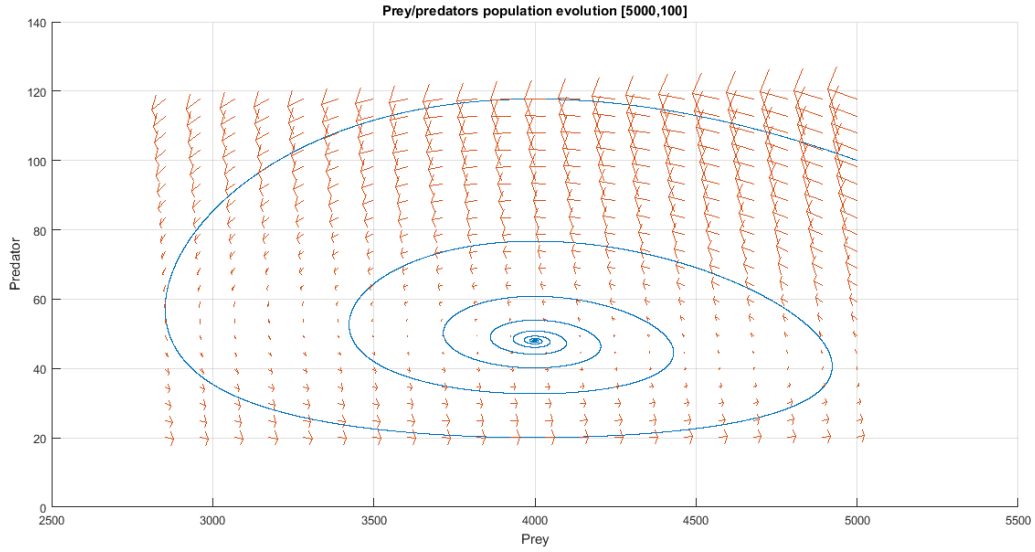
The $(\mathbf{1} - \frac{\boldsymbol{N_{prey}(t)}}{\boldsymbol{K}})$ **factor** replaces unrestricted prey growth by logistic equation-based model.

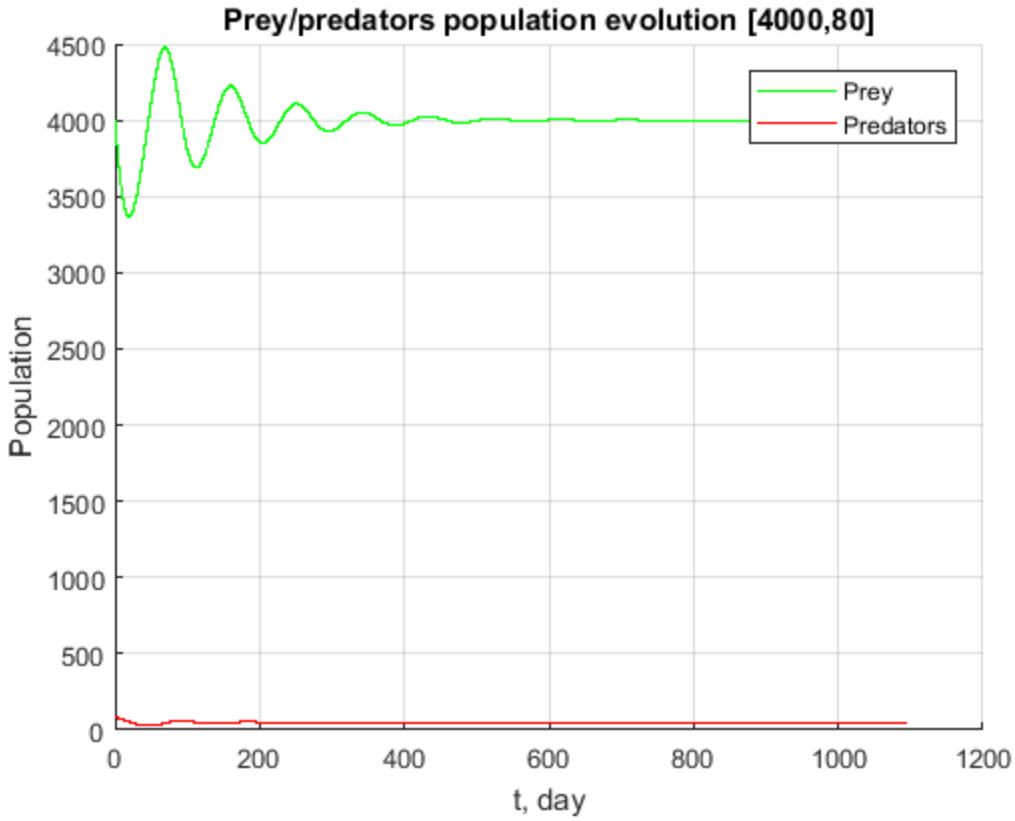
Assuming same parameters values and $K = 10^4$, simulation implemented by MATLAB script (the code is attached in appendix section – Appendix B) have given next results:

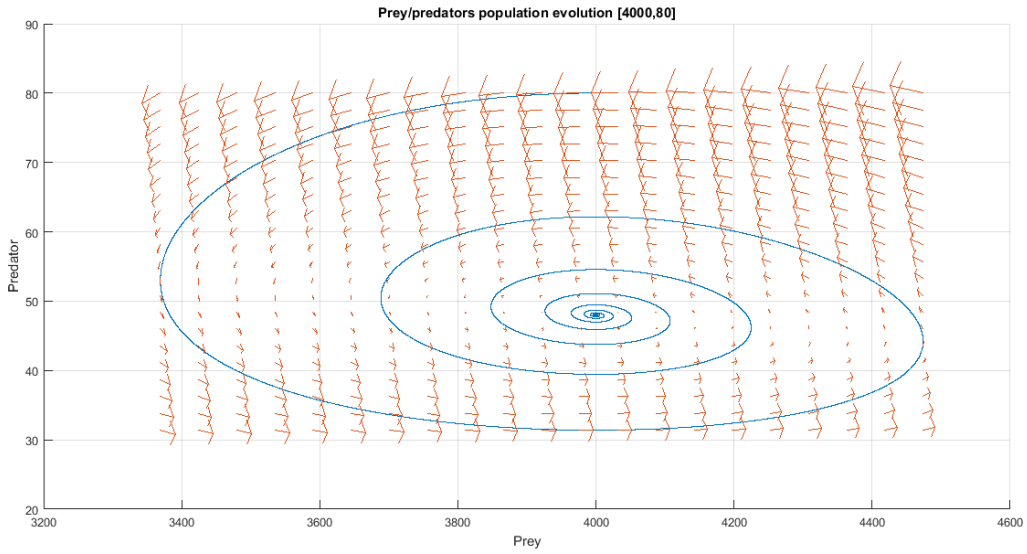



Prey/predators population evolution [200,50]
Predator
Prey
300
250
200
150
100
50
0
-50
0
1000
2000
3000
4000
5000
6000
7000
8000
9000


Prey/predators population evolution [5000,100]
Population
t, day
Prey
Predators
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200


Prey/predators population evolution [5000,100]
Predator
Prey
140
120
100
80
60
40
20
0
2500
3000
3500
4000
4500
5000
5500


Prey/predators population evolution [4000,80]
Prey
Predators
Population
t, day
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200



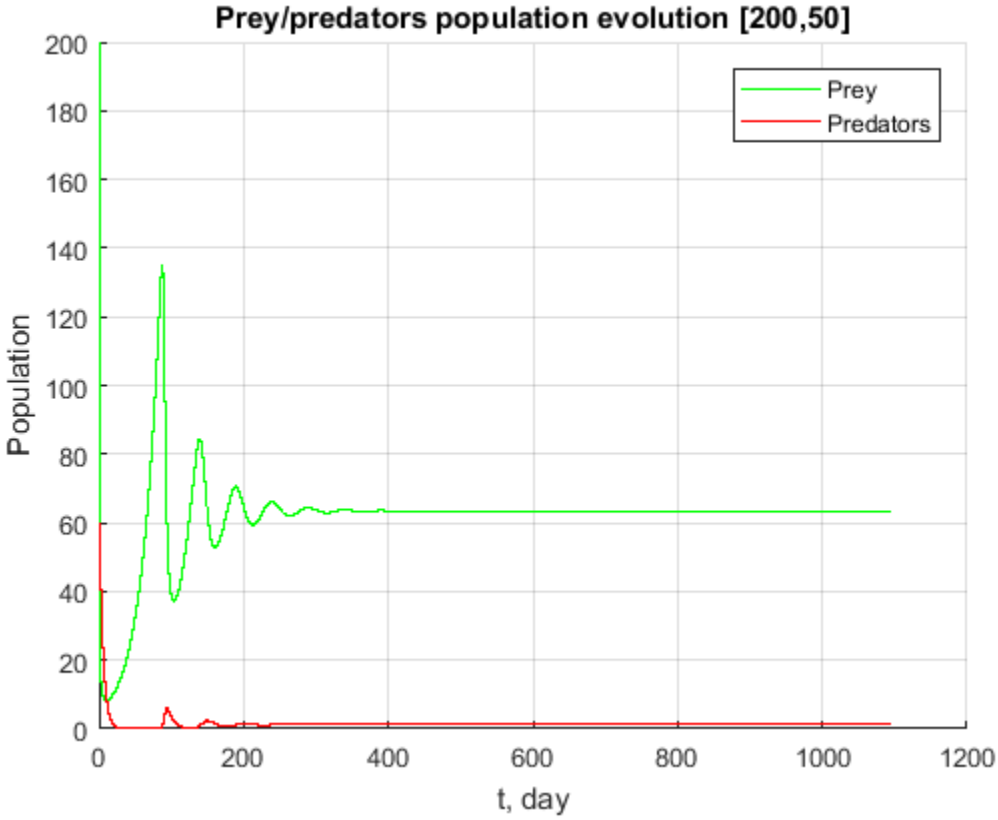
Finally, It was proposed to update the model as follows:

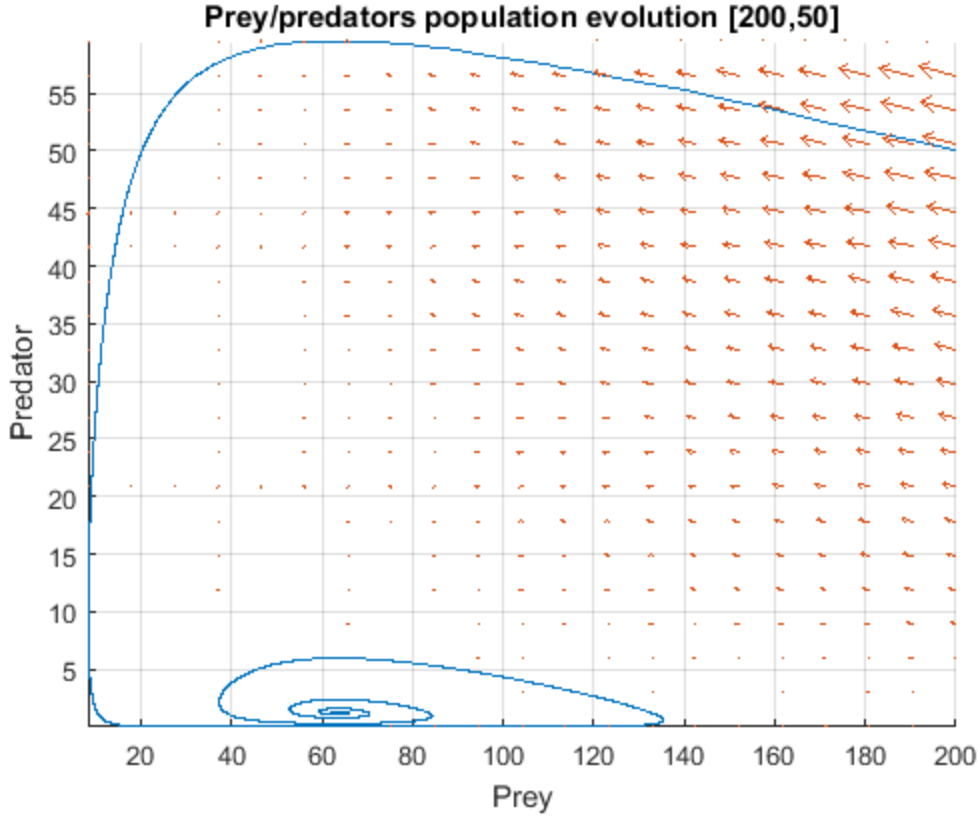
$$\frac{dN_{prey}}{dt} = R_{o,prey} \cdot (\mathbf{1} - \frac{\boldsymbol{N_{prey}(t)}}{\boldsymbol{K}}) \cdot N_{prey}(t) - (\frac{\boldsymbol{\gamma \cdot N_{prey}(t)}}{\mathbf{1} + \boldsymbol{\gamma \cdot h \cdot N_{prey}(t)}}) \cdot N_{prey}(t) \cdot N_{pred}(t)$$

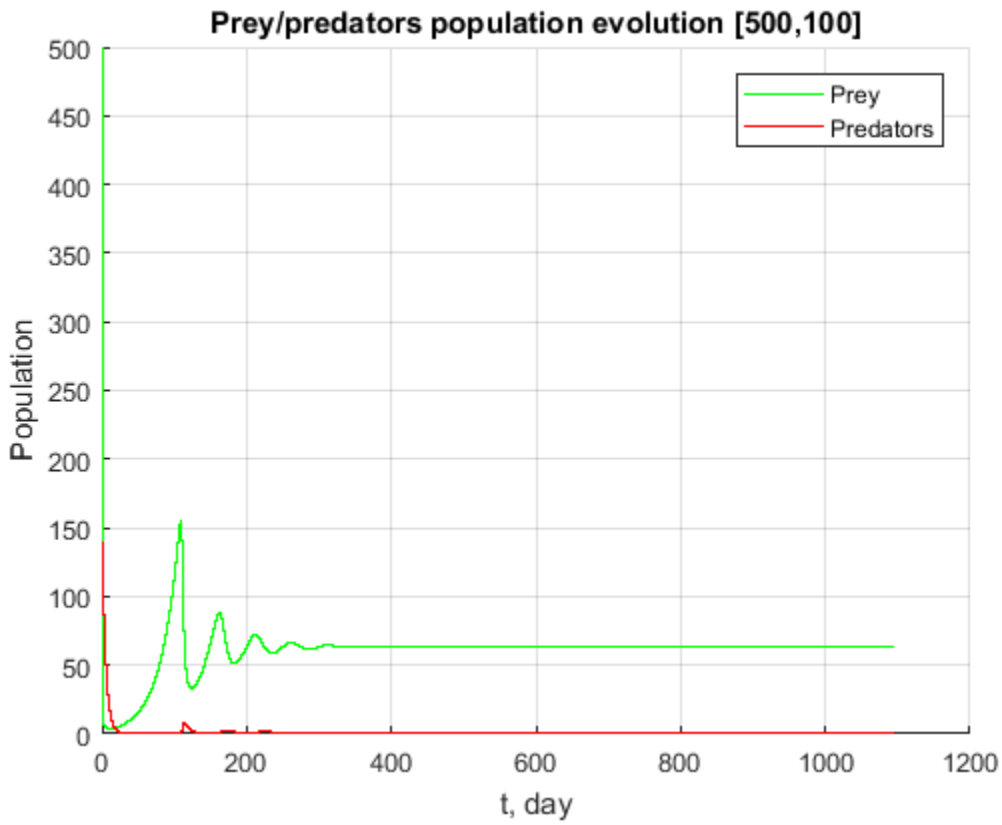
$$\frac{dN_{pred}}{dt} = \epsilon \cdot (\frac{\boldsymbol{\gamma \cdot N_{prey}(t)}}{\mathbf{1} + \boldsymbol{\gamma \cdot h \cdot N_{prey}(t)}}) \cdot N_{prey}(t) \cdot N_{pred}(t) - R_{o,pred} \cdot N_{pred}(t)$$

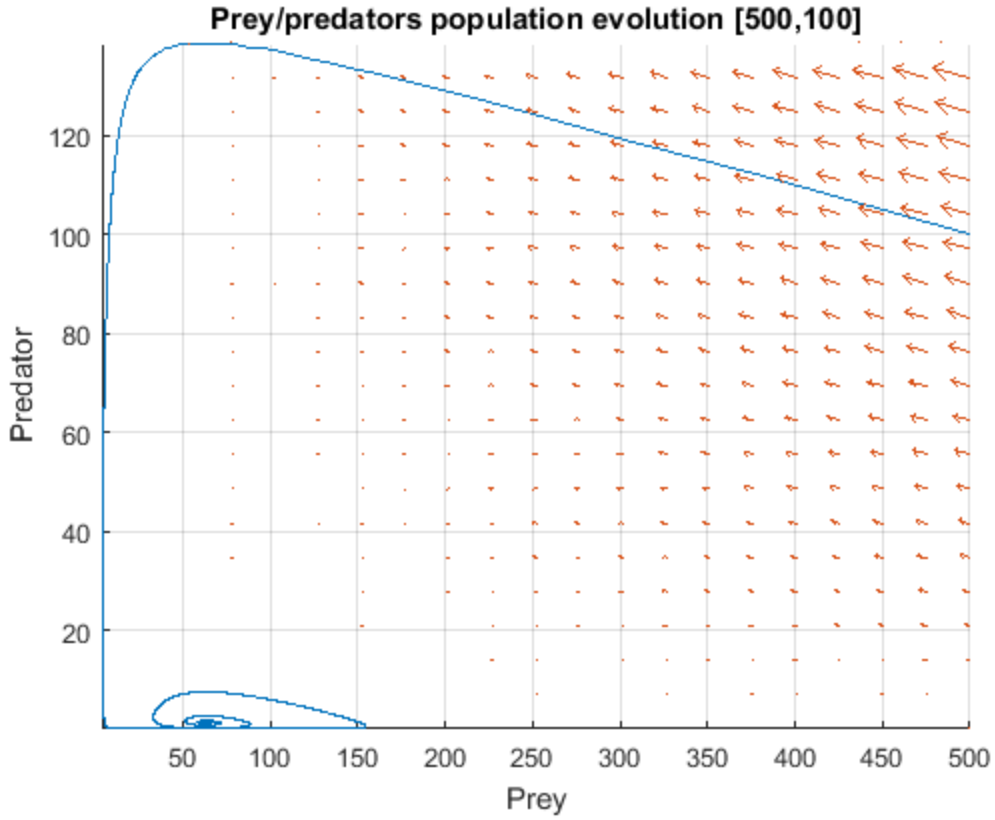
The $(\frac{\boldsymbol{\gamma \cdot N_{pred}(t)}}{\mathbf{1}+\boldsymbol{\gamma \cdot h \cdot N_{pred}(t)}})$ **factor** saturate predictor response at high prey density using Holling's disk equation..

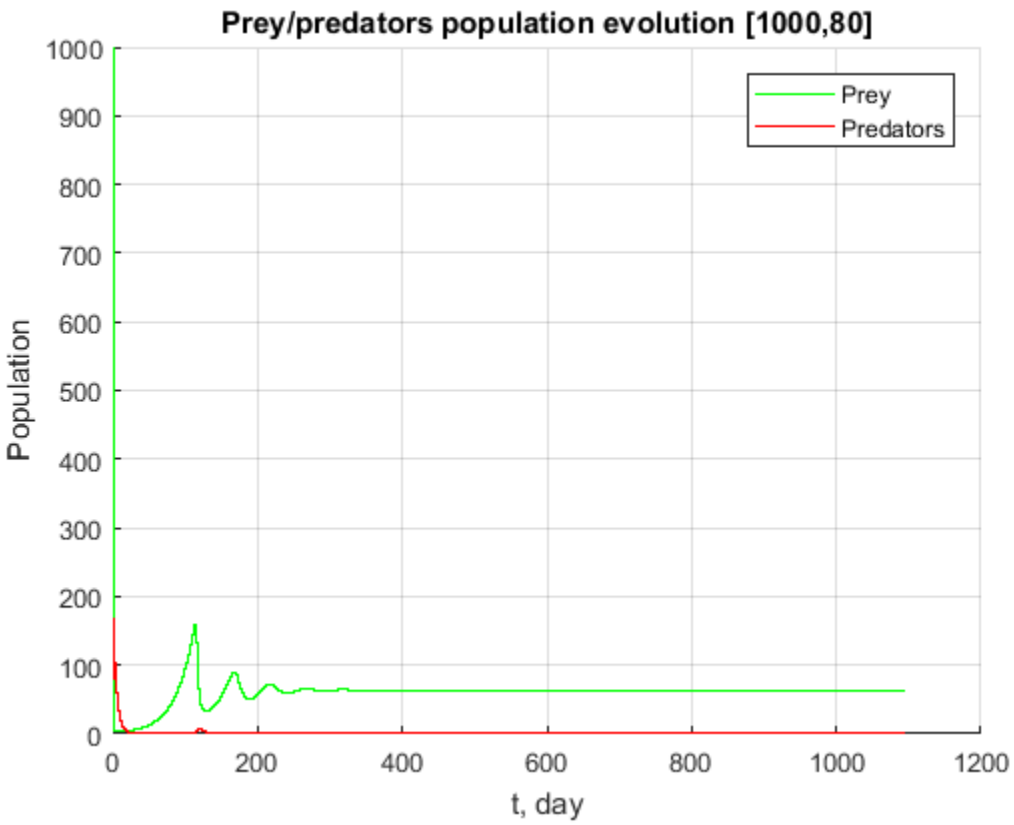
Assuming same parameters values and $h = 0.2$, simulation implemented by MATLAB script (the code is attached in appendix section – Appendix C) have given next results:

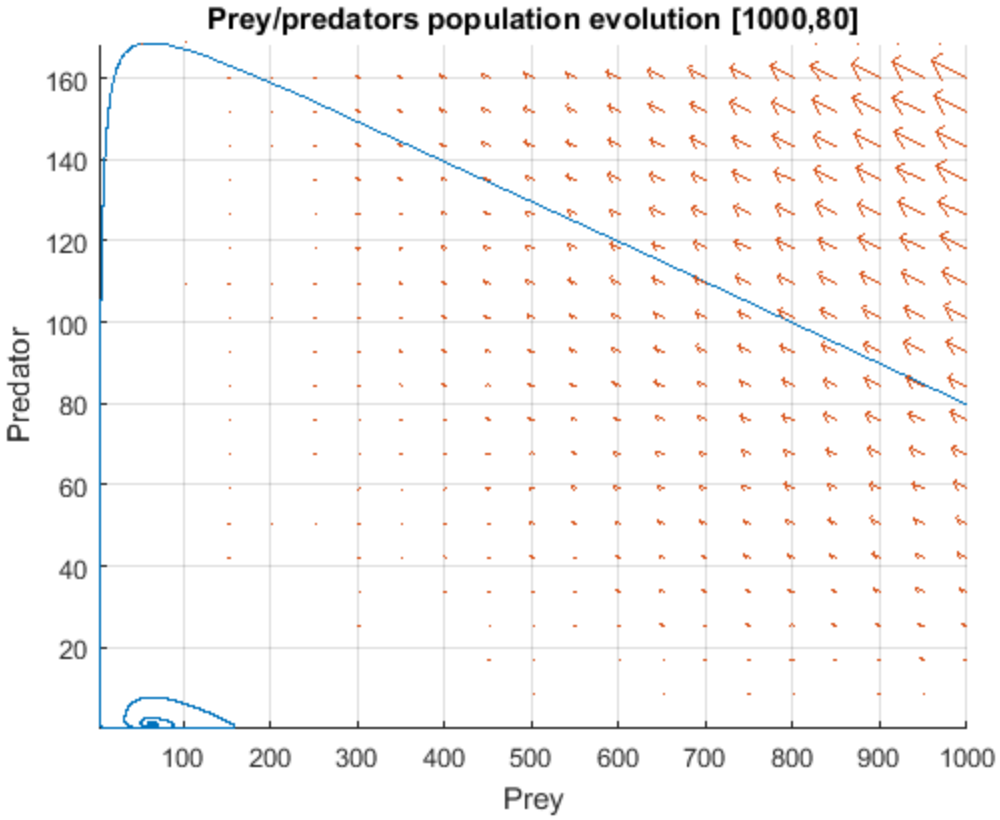

Prey/predators population evolution [200,50]
Prey
Predators
Population
t, day
0
20
40
60
80
100
120
140
160
180
200
0
200
400
600
800
1000
1200


Prey/predators population evolution [200,50]
Predator
Prey
5
10
15
20
25
30
35
40
45
50
55
20
40
60
80
100
120
140
160
180
200


Prey/predators population evolution [500,100]
Prey
Predators
Population
t, day
0
50
100
150
200
250
300
350
400
450
500
0
200
400
600
800
1000
1200


Prey/predators population evolution [500,100]
Predator
Prey
20
40
60
80
100
120
50
100
150
200
250
300
350
400
450
500


Prey/predators population evolution [1000,80]
Prey
Predators
Population
t, day
0
100
200
300
400
500
600
700
800
900
1000
0
200
400
600
800
1000
1200


Prey/predators population evolution [1000,80]
Predator
Prey
20
40
60
80
100
120
140
160
100
200
300
400
500
600
700
800
900
1000

Remark: it is possible to modify the general equation in different ways and possibly due to misunderstanding of the assignment instructions investigated here modifications differ from requested (the last modification seems a bit strange and so it is the source of anxiety). The link below (WolframAlpha) shows the solution “for equilibrium problem”:

https://www.wolframalpha.com/input/?i=%7B0.04*(1-x%2F10000)*x+-+0.0005*x*y%3D%3D0,+0.1*0.0005*x*y+-+0.2*y%3D%30%7D

The first modification just limit the prey population from the up – food to feed prey is limited and so at some population value there would not be enough food to feed so big prey population. The system with such modification still has stable point (focus that is shown on phase plane above). The solution

The second modification, possibly, define the prey defense possibilities – bigger herd can defense itself better then small. There are still few stable point: the first is obvious (no predators and maximum prey), but the second is for small population of both prey and predator. The link below (WolframAlpha) shows the solution “for equilibrium problem”

https://www.wolframalpha.com/input/?i=%7B0.04*(1-x%2F10000)*x+-+(0.0005*x%2F(1%2B0.0005*0.2*x))*x*y%3D%3D0,+0.1*(0.0005*x%2F(1%2B0.0005*0.2*x))*x*y+-+0.2*y%3D%3D0%7D

The results for first and the second modification have similarities – both have some point of equilibrium and the system go to this point through stable focus in contrast to the original equation system where are no “drag\friction” and so “limit cycles” occurs

Appendix A

part_a.m

```
%% Preparations
% Clear all: command window, workspace, figures
clc;
clear all;
close all;

% Define constants
R_o_prey = 0.04;
R_o_pred = 0.2;
y = 0.0005;
e = 0.1;

% Define the Lotka-Volterra model equations

dN_prey = @(N_prey, N_pred) R_o_prey.*N_prey - y*N_prey.*N_pred;
dN_pred = @(N_prey, N_pred) e*y*N_prey.*N_pred -R_o_pred.*N_pred;

% Define initial conditions
% (three cases)
N_prey_0 = [200 5000 4000];
N_pred_0 = [50  100  80];

% Define the time vector and array to hold the simulation results
dt = 0.01; time = 0:0.01:365;  % (year with 0.01 day time step)
N_prey = zeros(size(time,2),size(N_prey_0,2));
N_pred = zeros(size(time,2),size(N_pred_0,2));

% Initialization:
N_prey(1,:) = N_prey_0;
N_pred(1,:) = N_pred_0;

%% Solving
    % Use Euler method - step through all time steps
for i = 2:size(time,2)
    N_prey(i,:) = N_prey(i-1,:) +...
                  dt * dN_prey(N_prey(i-1,:),N_pred(i-1,:));
    N_pred(i,:) = N_pred(i-1,:) +...
                  dt * dN_pred(N_prey(i-1,:),N_pred(i-1,:));
end

%% Present the result

    % Plot simulated results for each initial condition.
for j = 1:size(N_prey_0,2)
    figure;
    hold on;
    grid on;
    plot(time, N_prey(:,j),'g');
    plot(time, N_pred(:,j),'r');
    hold off;
    xlabel('t, day');
```

```
    ylabel('Population');
    title(strcat('Prey/predators population evolution [',...
                  num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
    legend('Prey','Predators');

    % Plot the Pred-to-Prey population with derivative vectors
    figure;
    grid on;
    hold on;
        % Pred-to-Prey population (somekind of phase plane)
    plot(N_prey(:,j),N_pred(:,j));

        % Estimate the meshgrid for derivatives presenting
    [N_prey_mesh,N_pred_mesh] = meshgrid(min(N_prey(:,j)):((max(N_prey(:,j))-
min(N_prey(:,j)))/20):max(N_prey(:,j)),...
                                         min(N_pred(:,j)):((max(N_pred(:,j))-
min(N_pred(:,j)))/20):max(N_pred(:,j)));
        % Plot the vector field of derivatives
    quiver(N_prey_mesh,N_pred_mesh,...
           dN_prey(N_prey_mesh,N_pred_mesh),...
           dN_pred(N_prey_mesh,N_pred_mesh));
    hold off;
    xlabel('Prey');
    ylabel('Predator');
    title(strcat('Prey/predators population evolution [',...
                  num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
end
```

Appendix B

part_b_1.m

```
%% Preparations
% Clear all: command window, workspace, figures
clc;
clear all;
close all;

% Define constants
R_o_prey = 0.04;
R_o_pred = 0.2;
y = 0.0005;
e = 0.1;
K = 10000;

% Define the Lotka-Volterra model equations (modification #1)

dN_prey = @(N_prey, N_pred) R_o_prey*(1 - N_prey/K).*N_prey -
y*N_prey.*N_pred;
dN_pred = @(N_prey, N_pred) e*y*N_prey.*N_pred -R_o_pred.*N_pred;

% Define initial conditions
% (three cases)
N_prey_0 = [200 5000 4000];
N_pred_0 = [50  100  80];

% Define the time vector and array to hold the simulation results
dt = 0.01; time = 0:0.01:(3*365);  % (year with 0.01 day time step)
N_prey = zeros(size(time,2),size(N_prey_0,2));
N_pred = zeros(size(time,2),size(N_pred_0,2));

% Initialization:
N_prey(1,:) = N_prey_0;
N_pred(1,:) = N_pred_0;

%% Solving
    % Use Euler method - step through all time steps
for i = 2:size(time,2)
    N_prey(i,:) = N_prey(i-1,:) +...
                  dt * dN_prey(N_prey(i-1,:),N_pred(i-1,:));
    N_pred(i,:) = N_pred(i-1,:) +...
                  dt * dN_pred(N_prey(i-1,:),N_pred(i-1,:));
end

%% Present the result

    % Plot simulated results for each initial condition.
for j = 1:size(N_prey_0,2)
    figure;
    hold on;
    grid on;
    plot(time, N_prey(:,j),'g');
    plot(time, N_pred(:,j),'r');
```

```
    hold off;
    xlabel('t, day');
    ylabel('Population');
    title(strcat('Prey/predators population evolution [',...
                 num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
    legend('Prey','Predators');

    % Plot the Pred-to-Prey population with derivative vectors
    figure;
    grid on;
    hold on;
        % Pred-to-Prey population (somekind of phase plane)
    plot(N_prey(:,j),N_pred(:,j));

        % Estimate the meshgrid for derivatives presenting
    [N_prey_mesh,N_pred_mesh] = meshgrid(min(N_prey(:,j)):((max(N_prey(:,j))-
min(N_prey(:,j)))/20):max(N_prey(:,j)),...
                                         min(N_pred(:,j)):((max(N_pred(:,j))-
min(N_pred(:,j)))/20):max(N_pred(:,j)));
        % Plot the vector field of derivatives
    quiver(N_prey_mesh,N_pred_mesh,...
           dN_prey(N_prey_mesh,N_pred_mesh),...
           dN_pred(N_prey_mesh,N_pred_mesh));
    hold off;
    xlabel('Prey');
    ylabel('Predator');
    title(strcat('Prey/predators population evolution [',...
                 num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
end
```

Appendix C

part_b_2.m

```
%% Preparations
% Clear all: command window, workspace, figures
clc;
clear all;
close all;

% Define constants
R_o_prey = 0.04;
R_o_pred = 0.2;
y = 0.0005;
e = 0.1;
K = 10000;
h = 0.2;

% Define the Lotka-Volterra model equations (modification #2)

dN_prey = @(N_prey, N_pred) R_o_prey*(1 - N_prey/K).*N_prey -
(y*N_prey./(1+y*h*N_prey)).*N_prey.*N_pred;
dN_pred = @(N_prey, N_pred) e*(y*N_prey./(1+y*h*N_prey)).*N_prey.*N_pred -
R_o_pred.*N_pred;

% Define initial conditions
% (three cases)
N_prey_0 = [200 500 1000];
N_pred_0 = [50  100  80];

% Define the time vector and array to hold the simulation results
dt = 0.01; time = 0:0.01:(3*365);  % (year with 0.01 day time step)
N_prey = zeros(size(time,2),size(N_prey_0,2));
N_pred = zeros(size(time,2),size(N_pred_0,2));

% Initialization:
N_prey(1,:) = N_prey_0;
N_pred(1,:) = N_pred_0;

%% Solving
    % Use Euler method - step through all time steps
for i = 2:size(time,2)
    N_prey(i,:) = N_prey(i-1,:) +...
                  dt * dN_prey(N_prey(i-1,:),N_pred(i-1,:));
    N_pred(i,:) = N_pred(i-1,:) +...
                  dt * dN_pred(N_prey(i-1,:),N_pred(i-1,:));
end

%% Present the result

    % Plot simulated results for each initial condition.
for j = 1:size(N_prey_0,2)
    figure;
    hold on;
    grid on;
```

```
    plot(time, N_prey(:,j),'g');
    plot(time, N_pred(:,j),'r');
    hold off;
    xlabel('t, day');
    ylabel('Population');
    title(strcat('Prey/predators population evolution [',...
                 num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
    legend('Prey','Predators');

    % Plot the Pred-to-Prey population with derivative vectors
    figure;
    grid on;
    hold on;
        % Pred-to-Prey population (somekind of phase plane)
    plot(N_prey(:,j),N_pred(:,j));

        % Estimate the meshgrid for derivatives presenting
    [N_prey_mesh,N_pred_mesh] = meshgrid(min(N_prey(:,j)):((max(N_prey(:,j))-
min(N_prey(:,j)))/20):max(N_prey(:,j)),...
                                         min(N_pred(:,j)):((max(N_pred(:,j))-
min(N_pred(:,j)))/20):max(N_pred(:,j)));
        % Plot the vector field of derivatives
    quiver(N_prey_mesh,N_pred_mesh,...
           dN_prey(N_prey_mesh,N_pred_mesh),...
           dN_pred(N_prey_mesh,N_pred_mesh));
    hold off;
    xlabel('Prey');
    ylabel('Predator');
    xlim([min(N_prey(:,j)) max(N_prey(:,j))]);
    ylim([min(N_pred(:,j)) max(N_pred(:,j))]);
    title(strcat('Prey/predators population evolution [',...
                 num2str(N_prey_0(j)),', ',num2str(N_pred_0(j)),']'));
end
```