MH3110 Project: The Fault in Our Stars

Academic Year 2016/2017, Semester II

Abstract

We modeled two ordinary differential equations based on the movie 'The Fault in Our Stars'. The Lotka-Volterra model was proposed for the dynamics of love between Hazell and Augustus due to the oscillation of their feelings throughout the movie. Through this project, we attempted to predict the progress of their relationship in the long-run. Using the range between 0 and 20, we gave an average rating score of their emotions at different times of the movie and plotted the respective graphs to reflect the situation.

① Introduction

The movie 'The Fault in Our Stars' [3] revolves around the female lead, Hazell Grace Lancaster, her love life with Augustus Waters and the encounter with her favourite author, Peter Van Houten. Hazell and Augustus are a couple and met many obstacles to their love dynamics. During their overseas trip, Augustus's condition relapsed midway through the trip. This posed as a challenge to their relationship. The intention of this project is to determine the feelings they have for each other in the long run, assuming that Augustus is alive.

The classical Lotka-Volterra systems (which is used in this paper) models the densities of species in an ecosystem under the assumption that the growth rate of each species is a linear function of the abundances of the species in the system. This framework allows for the predator-prey, competitive, and symbiotic relationships to be modelled under Lotka-Volterra. However, the model does not take into account environmental factors such as diversity of interspecies relationships. [10]

Based on the movie being centered about love dynamics, we have referenced papers that have previously modelled love of two people using a system of differential equations [2,8,9]. For instance, the paper from Rinaldi (1998) [8] on love dynamics assumed the relationship between two parties to be of a linear form and interdependent on each other. A model using a system of Ordinary Differential Equations using Lotka-Voterra models. The result obtained was a stable equilibrium. After analysis of the data obtained, we found that our model is closely related to that was presented by Rinaldi and proceeded to adopt the Lotka-Volterra model for the two characters.

Using information from Hussein (2013)[4], we were able to model our system accurately to fit the context of our research. Employing the Particle Swarm Optimisation algorithm[6], which is an evolution based algorithm based on swarm intelligence, we were able to optimise our differential equations. On every iteration of the optimisation algorithm, each data point in the set has a fitness value that is assigned through computation of the optimisation function and this value evaluates the feasibility of the particle at the current position. As such, we were able to predict the relationship between the two characters in the long run.

However, the mentioned model does not take into account environmental changes that may affect the relationship between Augustus and Hazell which is similar to the limitations that Rinaldi (1998)[8] had previously stated, where "aging, learning and adaptation processes" may affect the eventual outcome. Future research on topics relating to modelling love using differential equations can be conducted while taking into account the previously mentioned excluded factors to provide greater accuracy in predicting the relationship between two parties in the future.

② Data Collection and Technique

The data was collected throughout the movie by assessing their emotions on a scale between 0 and 20, with neutral being 10. The Lotka-Volterra model requires that we rate the emotions starting from 0 as Lotka-Volterra is used to describe the dynamics of related biological systems, indicating that the values used in the model must be non-negative so that the use of the model is justified. The ratings obtained were based on the state of emotions towards the other person and their countenance as well. The following is an extract of the data that was collected:

Time (min)	Hazell→Augustus	Augutus→Hazell
7.03	10.0	11.0
12.30	12.2	15.5
86.32	14.7	10.0

At 7.03 minutes, the rating of Hazell's emotions towards Augustus remains at 10 as it is the neutral point. This is the point where Hazell just met Augustus, where she does not yet have any feelings for Augustus At 12.30 minutes, Hazell's emotions increased to 12.2 from the initial 10 due to her interaction with Augustus and having a good impression of him. At 86.32 minutes, Augustus passed away after his cancer relapsed resulting in the point dropping back to 10 (neutral point) as his feeling is non-existent.

The following is a plot of all the data points[1] that were collected from the movie:

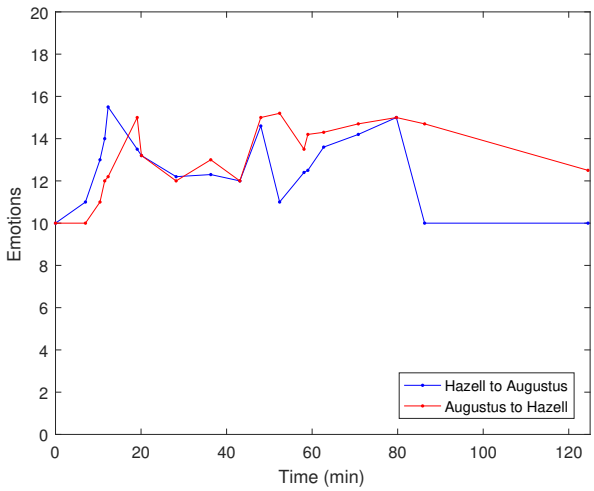

Figure 1: Love dynamics between Hazell and Augustus

③ Modelling

3.1 Lotka-Volterra Model

The Lotka-Volterra model, also known as the predator-prey model, is usually used to model the ecology of how the predator and prey interact with each other. As the shape of the graph is similar to the Lotka-Volterra model, we will be emulating the model to for our system of ordinary differential equations. In the model, the predator usually lags behind the prey, so we can assume Hazell's feelings towards Augustus to be the "predator" and Augustus' feelings towards Hazell to be the "prey".

We assume that their feelings are only affected by their own emotions and the other party's behaviour. We excluded external factors, such as family and health condition before proceeding to derive a system of ordinary differential equations as shown below:

$$\frac{dA}{dt} = \alpha_1 \cdot A - \alpha_2 \cdot A \cdot H$$

$$\frac{dH}{dt} = \beta_1 \cdot A \cdot H - \beta_2 \cdot H$$

A: The level of affection that Augustus has towards Hazell at time t
H: The level of affection that Hazell has towards Augustus at time t
$A \cdot H$: Effects on relationship due to the other party's actions
$\alpha_1, \alpha_2, \beta_1, \beta_2$: positive constants describing the love dynamics between each other

The initial values of A and H were observed at the start of the movie and defined to be neutral, $\therefore H(0) = 10$ and $A(0) = 10$.

3.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a stochastic optimisation technique originally intended for emulating the social behaviour of bird flocks or fish schools. This method works by iteratively trying to improve a candidate solution within a given measure of quality. This problem is solved by having a large number of particles move around within a defined space, also known as a *search space*. Each particle is affected by its best-known position, current position and velocity. These factors are further constrained by the proposed model and its search space. The movement of the swarm of particles is expected to converge towards the best solution over multiple iterations.

To determine the search space for the current scenario, we must consider our proposed model and determine the domain for the functions $A(t)$ and $H(t)$. Since we are using the Lotka-Volterra model and have bounded the 2 functions above by 20, our domain for $A(t)$ and $H(t)$ is $[0, 20]$.

3.3 Fitting Parameters of the model into data

To obtain the coefficients $\alpha_1, \alpha_2, \beta_1, \beta_2$, we used the Least Square Method by estimating $A'(t)$ and $H'(t)$ based on the collected data points[1].

$$\text{Least Squares Error (LSE): } E(\alpha_1, \alpha_2, \beta_1, \beta_2) = \sum_{i=1}^{n} (M(t) - M(t_i, \alpha_1, \alpha_2, \beta_1, \beta_2))^2,$$

As Augustus and Hazell are a couple and Augustus is more emotionally susceptible when Hazell faces any issues, he is more likely to be affected by the relationship. This is reflected by α_2 having a higher numerical value than β_1. Augustus initiated the conversation with Hazell, so it shows that Augustus has greater affection for Hazell. This is reflected by β_2 having a smaller numerical value than α_1. When Hazell's condition deteriorated, she avoided Augustus as she does not want to implicate Augustus and is the reason that β_1 is smaller β_2.

Performing PSO using the Lotka-Volterra models and considering the minimization of the LSE, we obtain the following optimisation plots:

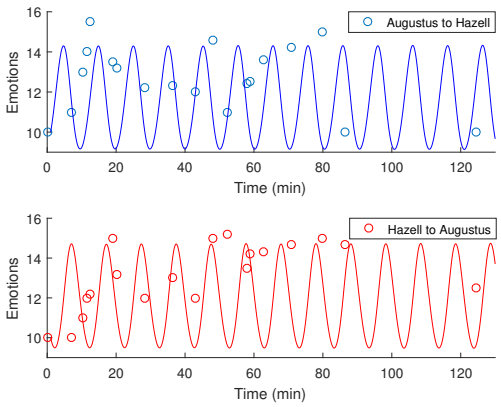

Figure 2: Separate Lotka-Volterra Models

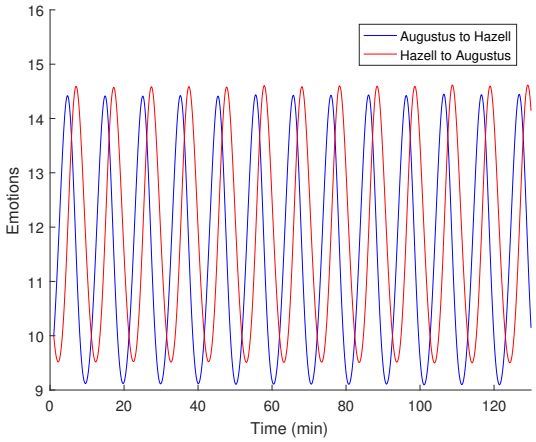

Figure 3: Combined Lotka-Volterra Models

After implementing the PSO yields the following:

$$\alpha_1 = 0.6659 \quad \alpha_2 = 0.0561 \quad \beta_1 = 0.0499 \quad \beta_2 = 0.5770$$

4 Analysis of Model

4.1 Obtaining Eigenvalues

Substituting the obtained variables into the model gives us the love dynamics between Hazell and Augustus as follows:

$$\frac{dA}{dt} = 0.6659 \cdot A - 0.0561 \cdot A \cdot H$$
$$\frac{dH}{dt} = 0.0499 \cdot A \cdot H - 0.5770 \cdot H$$

Both $A(t)$ and $H(t)$ has initial values 10 at $t = 0$ as the feelings of Hazell and Augustus towards each other were neutral in the beginning.

The model has been found to have two critical points $A = 0, H = 0$ or $A = 11.5631, H = 11.8698$.
The 2×2 Jacobian matrix for the model is:

$$J(A, H) = \begin{bmatrix} 0.6659 - 0.0561H & -0.0561A \\ 0.0499H & 0.0499A - 0.5770 \end{bmatrix}$$

At the critical point (0,0),

$$J(0, 0) - \lambda I_2 = \begin{bmatrix} 0.6659 - \lambda & 0 \\ 0 & -0.5770 - \lambda \end{bmatrix}$$

$\lambda = 0.6659$ or $\lambda = -0.5770$

At the critical point (11.5631,11.8698),

$$J(11.5631, 11.8698) - \lambda I_2 = \begin{bmatrix} 1.28 \cdot 10^{-9} - \lambda & -0.648691383 \\ 0.592306774 & -1.25 \cdot 10^{-10} - \lambda \end{bmatrix}$$

$\lambda = \pm 0.619858i$

4.2 Plotting Phase Portraits

From the eigenvalues obtained, we can subsequently plot the respective phase portraits as shown in Figure 4 and 5. When t tends to ∞, the critical point at:

1. $(0, 0)$ forms an unstable saddle point (Figure 4).
2. $(11.5631, 11.8698)$ forms a Lyapunov stable centre (Figure 5).

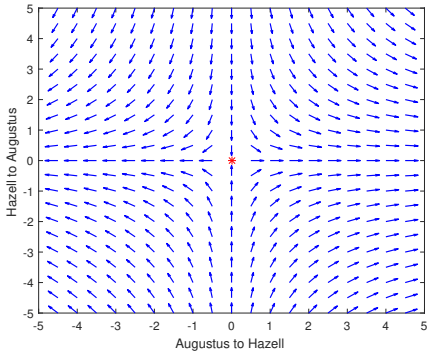

Figure 4: Critical Point (0,0)

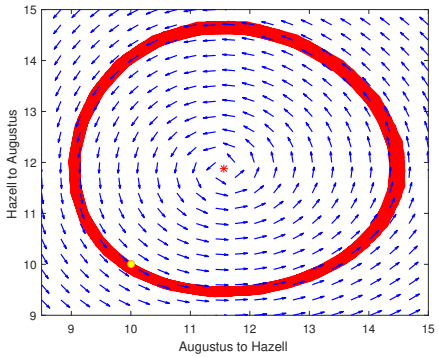

Figure 5: Critical Point (11.5631,11.8698)

(Remark: The red star represents the critical point of the system and the yellow circle represents the initial point of the system of differential equations.)

5 Advanced Mathematical Analysis

Definition (Lyapunov function)
Consider the system of differential equations **F**

$$\frac{dx}{dt} = f(x) \tag{1}$$

where $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^n$ *is continuous. We call* V *a Lyapunov function on* $G \subseteq \Omega$ *for the system* (1) *if*

(i) V *is continuous on* G,

(ii) *If* V *is not continuous at* $\overline{x} \in \overline{G}$ *(the closure of* G*) then* $\lim_{\substack{x \to \overline{x} \\ x \in G}} V(x) = +\infty$,

(iii) $\dot{V} = \nabla V \cdot f \leq 0$ *on* G

Theorem (Lyapunov stability)
Let $U \subseteq \mathbb{R}^n$ *be open and* $f : U \to \mathbb{R}$ *be continuously differentiable and such that* $f(x_0) = 0$ *for some* $x_0 \in U$. *Suppose further that there is a real-valued function* $V : U \to \mathbb{R}$ *that satisfies*

(i) $V(x_0) = 0$

(ii) $V(x) > 0$ *for* $x \in U \backslash \{x_0\}$

Then if

(a) $\dot{V}(x) \leq 0 \quad \forall x \in U$ *then* x_0 *is Lyapunov stable*;

(b) $\dot{V}(x) < 0 \quad \forall U \backslash \{x_0\}$ *then* x_0 *is asymptotically stable*;

(c) $\dot{V}(x) > 0 \quad \forall x \in U \backslash \{x_0\}$ *then* x_0 *is unstable.*

Theorem (LaSalle's Invariance Principle)
Assume that V is a Lyapunov function of $\boldsymbol{F}$ *on G. Define* $S = \{x \in \overline{G} \cap \Omega : \dot{V}(x) = 0\}$. *Let M be the largest invariant set in S. Then every bounded trajectory (for* $t \geq 0$*) of* $\boldsymbol{F}$ *that remains in G approaches the set M as* $t \to +\infty$.

Proof. By considering the equations:

$$\frac{dA}{dt} = \alpha_1 \cdot A - \alpha_2 \cdot A \cdot H \qquad \frac{dH}{dt} = \beta_1 \cdot A \cdot H - \beta_2 \cdot H \tag{2}$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2 > 0$ and $A(0) > 0, H(0) > 0$

Take $A^* = \frac{\beta_2}{\beta_1}, H^* = \frac{\alpha_1}{\alpha_2}, E = (A^*, H^*)$. Then it follows that

$$\frac{dA}{dH} = \frac{\frac{dA}{dt}}{\frac{dH}{dt}} = \frac{-\alpha_2 \cdot A(H - H^*)}{\beta_1 \cdot H(A - A^*)}$$

As the equation is a first-order separable ODE, we can manipulate the fraction to obtain the following:

$$\frac{A - A^*}{A} dA + \frac{\alpha_2}{\beta_1} \frac{H - H^*}{H} dH = 0 \tag{3}$$

We can introduce the following equation for our Lotka-Volterra equation:

$$V(A, H) = \int_{A^*}^{A} \frac{\eta - A^*}{\eta} d\eta + \frac{\alpha_2}{\beta_1} \int_{H^*}^{H} \frac{\psi - H^*}{\psi} d\psi \qquad (A, H) \neq (0, 0) \tag{4}$$

Remark: It can be proven that V is a Lyapunov function by the definition as $V(A, H)$ is the integration of an elementary function.

Using (3) and (4), we obtain

$$\dot{V}(A, H) = \frac{dV}{dt} = \frac{A - A^*}{A} \frac{dA}{dt} + \frac{\alpha_2}{\beta_1} \frac{H - H^*}{H} \frac{dH}{dt} \equiv 0$$

Furthermore, using the fact that $V(A(t), H(t)) \equiv V(A(0), H(0)) \equiv C$ and using the Initial Values (IVP), we can conclude that the system (2) has a periodic solution. This also shows that every bounded trajectory of (2) remains in the set as t tends to ∞.

By the Lyapunov stability theorem, we can conclude that since $\dot{V}(x) = 0$, the system at the critical point (11.5631, 11.8698) is Lyapunov stable. Constructing a Lyapunov function to obtain stability of critical points does not require any numerical solutions. Additionally, this shows that there exists alternative methods (apart from using eigenvalues, *tr(J(A,H))* and *det(J(A,H))* in determining the stability of a critical point. However, there is no proven method to formulate a Lyapunov function for any given ODE model. It is therefore difficult to apply the Lyapunov theorem in general to any system.

(6) Conclusion

Based on the ending of the movie, we predict that the periodic oscillations would continue beyond the duration of the movie. The numerical simulation as depicted in Figure 3 supports this theory where the trend of emotions in the future would be similar to the prediction which has been previously stated.

Hazell continues to face the risk of her cancer relapsing and it may affect her attitude towards Augustus. However, this is based on our assumption that Augustus is still alive, which is essentially impossible based on the flow of events in the movie. In addition, we have to note the limitations of the model presented due to the assumptions made when implementing the model. The first is that it is difficult to measure the emotions purely based on numerical assessment.Secondly, we may be unable to provide a good estimation in the long run due to the factors as environmental and personality changes. This implies

that the solution curves and trajectory may provide a good estimation for the upcoming few months of their relationship but not in the distant future. Lastly, due to the complex competitive relationship between both characters in the movie, it may be difficult to predict how other characters will react due to the complexity of human emotions.

Nevertheless, the report serves as a reminder that there is no smooth-sailing relationship and there are bound to be highs and lows in relationships due to the effect external circumstances have on our emotions. Future research taking into account external factors such as including third party emotions (i.e. their parents opinions on their relationship) could provide a better assessment on the future relationship between Hazell and Augustus.

References

[1] Anisiu M. (2014). Lotka, Volterra And Their Model. *Didactica Mathematica, 32*, 9-17.

[2] Bielczyk, N., Bodnar, M., & Foryś, U. (2012). Delay can stabilize: Love affairs dynamics. *Applied Mathematics and Computation, 219*(8), 3923-3937. doi:10.1016/j.amc.2012.10.028

[3] Green, J. (2015). *The fault in our stars.* London: Penguin.

[4] Hussein, S. (2013). Predator-Prey Modeling. *Undergraduate Journal of Mathematical Modeling: One Two, 3*(1). doi:10.5038/2326-3652.3.1.32

[5] Hsu, S.B. (2005). A survey of constructing Lyapunov functions for mathematical models in population biology. *Taiwanese J. Math, 9*(2), 151-173.

[6] Pant, S., Kumar, A., Suraj Bhan, S., & Ram, M. (2017). A modified particle swarm optimization algorithm for nonlinear optimization. *Nonlinear Studies, 24*(1), 127-138. Retrieved from http://nonlinearstudies.com/index.php/nonlinear/article/view/1469

[7] Rachel, V. (2013). Predator Prey Models in Competitive Corporations. *Honors Program Projects.* 45.

[8] Rinaldi, S. (1998). Love dynamics: The case of linear couples. *Applied Mathematics and Computation, 95*(2-3), 181-192. doi:10.1016/s0096-3003(97)10081-9

[9] Rinaldi, S. (1998). Laura and Petrarch: An Intriguing Case of Cyclical Love Dynamics. *SIAM Journal on Applied Mathematics, 58*(4), 1205-1221. doi:10.1137/s003613999630592x

[10] Zhang, Y., Wang, S., & Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering, 2015,* 1-38. doi:10.1155/2015/931256

Appendix A: Data points

Time (min)	Hazell→Augustus	Augutus→Hazell
0.00	10.0	10.0
7.03	10.0	11.0
10.35	11.0	13.0
11.53	12.0	14.0
12.30	12.2	15.5
19.05	15.0	13.5
20.12	13.2	13.2
28.18	12.0	12.2
36.30	13.0	12.3
43.07	12.0	12.0
48.00	15.0	14.6
52.37	15.2	11.0
58.07	13.5	12.4
59.01	14.2	12.5
62.73	14.3	13.6
79.65	15.0	15.0
86.32	14.7	10.0
124.54	12.5	10.0

Appendix B: MATLAB Code

Lotka-Volterra and Particle Swarm Optimisation Code

```
clear all
close all

% Define data
load('ODEProj.mat')
idx = length(Lotka_Volterra_Data(:,1));
mytime = (1:idx)';
mydata(:,1) = Lotka_Volterra_Data(:,2);
mydata(:,2) = Lotka_Volterra_Data(:,3);

%Least square function
objfun = @(x) least_squares(x,mydata, mytime);

%% Optimisation using PSOPC with the same upper/lower bounds used in
%% simplex method
[k least_squares] = PSOPC(objfun, 4, [0.4 0 0.4 0], [0.8 0.8 0.8 0.8],
    1000)

%% Plot model with estimated parameters
y0(1) = 10.0; y0(2) = 10.0;
moviedur=[1:0.1:130];
[t,y] = ode45(@Lotka_Volterra_Model,moviedur,y0,[],k);

%Timestamp of important points
x = Lotka_Volterra_Data(:,1);

%Subplots for separate PSO VL graphs
subplot(2,1,1)
hold on
plot(x,mydata(:,1),'O');
plot(t,y(:,1),'-b')
axis([0 130 9 16]);
legend('Augustus to Hazell','location','northeast'); xlabel('Time (min)
    '); ylabel('Emotions');

subplot(2,1,2)
hold on
plot(x,mydata(:,2),'rO');
plot(t,y(:,2),'-r');
axis([0 130 9 16]);
legend('Hazell to Augustus','location','northeast'); xlabel('Time (min)
    '); ylabel('Emotions');

%Combined PSO VL graph
figure
hold on
plot(t,y(:,1),'-b');
plot(t,y(:,2),'-r');
legend('Augustus to Hazell','Hazell to Augustus','location','northeast'
    );
axis([0 130 9 16]);
xlabel('Time (min)');
ylabel('Emotions');
```

Least Square Error Calculation

```
function val=least_squares(k, mydata, mytime)

%Define initial values
y0(1) = 10.0; y0(2) = 10.0;
modelfun=@(t,x)Lotka_Volterra_Model(t,x,k);
[t ycalc]=ode45(modelfun,mytime,y0);
resid = (ycalc-mydata).*(ycalc-mydata);
val = sum(sum(resid));
end
```

Particle Swarm Optimisation Code (Courtesy of University of Liverpool)

```
function [bestparticle, fbestval] = PSO(fname, NDim,lbound, ubound,
    MaxIter)

% function [fbestval,bestparticle] = PSOPC(fname,bound,vmax,NDim,
    MaxIter)
%
%   Run a PSO with Passive Congregation (PSOPC) algorithm
%
%Input Arguments:
%   fname       - the name of the evaluation .m function
%   NDim        - dimension of the evalation function
%   MaxIter     - maximum iteration


% Modified Particle Swarm Optimization for Matlab
% Copyright (C) 2003 Shan He, the University of Liverpool
% Intelligence Engineering & Automation Group
% Last modifed 13-Aug-03

ploton=0;
figure

flag=0;
iteration = 0;
%MaxIter =1000;  % maximum number of iteration
PopSize=100;     % population of particles
%NDim=10;         % Number of dimension of search space
c1 = .6;        % PSO parameter C1
c2 = .6;        % PSO parameter C2
w=0.8;           % Inertia weigth
% decrease the inertia
startwaight = 0.9;
endwaight = 0.5;
waightstep = (startwaight-endwaight)/MaxIter;

c3step = (1 - 0.6)/MaxIter;


% Defined lower bound and upper bound.
LowerBound = zeros(NDim,PopSize);
UpperBound = zeros(NDim,PopSize);
for i=1:PopSize
    LowerBound(:,i) = lbound';
```

```
    UpperBound(:,i) = ubound';
end


DResult = 0;     % Desired results
population =  rand(NDim, PopSize).*(UpperBound-LowerBound) + LowerBound;     % Initialize swarm population
vmax = ones(NDim,PopSize);

for i=1:NDim
    vmax(i,:)=(UpperBound(i,:)-LowerBound(i,:))/2;
end
velocity = vmax.*rand(1);      % Initialize velocity

% Evaluate initial population
for i = 1:PopSize,
    fvalue(i) = feval(fname, (population(:,i)));
end

pbest = population;   % Initializing Best positions   matrix
fpbest = fvalue;      % Initializing the corresponding function values
% Finding best particle in initial population
[fbestval,index] = min(fvalue);    % Find the globe best
[fsortval, sortindex] = sort(fvalue);
while(flag == 0) & (iteration < MaxIter)
    iteration = iteration +1;
    w = startwaight - iteration*waightstep;
    for i=1:PopSize
        gbest(:,i) = population(:,index);
    end

    for i=1:PopSize
        rparticle(:,i) = population(:,floor(PopSize*rand(1))+1);
    end

    R1 = rand(NDim, PopSize);
    R2 = rand(NDim, PopSize);
    R3 = rand(NDim, PopSize);

    c3 = 0.6 + c3step*iteration;
    stationary=ones(NDim, PopSize);
    stationary(:,index)=0;
    sortmatrix = repmat(sortindex, NDim, 1)./PopSize;
    velocity = w*velocity + c1*R1.*(pbest-population) + c2*R2.*(gbest-population) + c3*R3.*sortmatrix.*(rparticle-population);
    % Update the swarm particle
    population = population + velocity;


    % Prevent particles from flying outside search space
    OutFlag = population<=LowerBound | population>=UpperBound;
    population = population - OutFlag.*velocity;

    % Evaluate the new swarm
    for i = 1:PopSize,
        fvalue(i) = feval(fname, (population(:,i)));
    end
    % Updating the pbest for each particle
```

```
    changeColumns = fvalue < fpbest;
    pbest(:, find(changeColumns)) = population(:, find(changeColumns));
            % find(changeColumns) find the columns which the values are
        1
    fpbest = fpbest.*( ~changeColumns) + fvalue.*changeColumns;
                % update fpbest value if fvalue is less than fpbest


    % Updating index
    [fbestval, index] = min(fvalue);
    [fsortval, sortindex] = sort(fvalue);

     % plot best fitness
     %hold on;
     Best(iteration) =fbestval;
     semilogy(Best,'r--');xlabel('generation'); ylabel('f(x)');
     text(0.5,0.95,['Best = ', num2str(Best(iteration))],'Units','
        normalized');
     drawnow;
end

bestparticle = population(:,index)
```

Phase Portrait

```
clc
clear all
%variable coefficients change the coefficients here
a1 = 0.6659;
a2 = 0.0561;
b2 = 0.5770;
b1 = 0.0499;


f = @(t,x) [a1*x(1) - a2*x(2)*x(1); b1*x(1)*x(2)- b2*x(2) ]; %ODE
    system
p = 7; %x-axis limit
q = 17; % y-axis limit
y1 = linspace(p,q,(q-p)*2);
y2 = linspace(p,q,(q-p)*2);

[x,y] = meshgrid(y1,y2);

u = zeros(size(x));
v = zeros(size(x));

t = 0;
for i = 1:numel(x)
    Yprime = f(t,[x(i);y(i)])
    u(i) = Yprime(1);
    v(i) = Yprime(2);
end

%plotting stuffs
quiver(x,y,u,v,'r');
figure(gcf);
xlim([p q])
```

```
ylim([p q])
grid on;

hold on
y20 = [10]
[ts ys] = ode45(f,[0 80],[y20 ; y20]);
plot(ys(:,1),ys(:,2))
```

Calculation of Variables

```
function dy = Lotka_Volterra_Model(t,y,k)
% Lotka-Volterra predator-prey model.
dy = zeros(2,1);
alpha = k(1); beta = k(2); gamma = k(3); delta = k(4);
A = [alpha - beta*y(2), 0; 0, ...
    -gamma + delta*y(1)];
dy = A*y;
end
```