Report on

# Assignment 4: Non-Linear ODE Solution Schemes Based on Implicit and Explicit Euler Methods

Submitted in partial fulfillment of the requirements of
ENM 502
Numerical Methods and Modeling
by

**Jalaj Maheshwari**
M.S.E., Mechanical Engineering and Applied Mechanics

University of Pennsylvania
School of Engineering and Applied Sciences

12th April, 2016

## Table of Contents

## Table of Figures

## 1. Introduction

The problem statement of this particular project deals with setting up non-linear ODE solution schemes based on both the implicit and explicit Euler methods for integrating the Lotka-Volterra system of ordinary differential equations representing a 'predator-prey' system given by Equation 1.

$$\frac{dx}{dt} = (a - by)x - px^2 \tag{1a}$$

$$\frac{dy}{dt} = (cx - d)y - qy^2 \tag{1b}$$

The parameters $\{a,b,c,d>0\}$ represent the natural prey growth rate, predator-prey interaction on prey death, the predator-prey interaction on predator growth, and the natural predator death rate, respectively. The "modified" refers to the last two terms, $px^2$ and $qy^2$ which can be thought of as the effects of overcrowding (or equally, competition) within a species, leading to a decline in the population (i.e. $p,q>0$).

The stationary points of this non-linear system of ODEs are found as a function of $\{a,b,c,d,p,q\}$. Non-linear ODE solution schemes based on both the implicit and explicit Euler methods for integrating these equations for given values of all the constants are set up. Using the parameter set ($a=b=c=d=1$ and $p=q=0$), an accuracy analysis of your ODE solvers by varying the time step size is performed. The Lotka-Volterra model is analyzed using phase plots for the parameters mentioned above.

## 2. Problem Setup and Formulation

A given ordinary differential equation represented by Equation 2, gives us a system of ODEs. For the given Lotka-Volterra problem, we have a system of non-linear ODEs. In order to solve this system of ordinary differential equations, we use implicit and explicit Euler methods.

$$\frac{d\underline{y}}{dt} = f(\underline{y}) \tag{2}$$

In the case of the explicit Euler method, if the solution $\underline{y_n}$ at a given instant is known, then the solution at $\underline{y_{n+1}}$ can be found by using Equation 3.

$$\underline{y_{n+1}} = h * \underline{f}\left(\underline{y_{n+1}}\right) + \underline{y_n} \tag{3}$$

where, '*h'* is the chosen time step size. If the initial condition $\underline{y_0}$ is known, the solutions at any further time instants can be found using the above equation by choosing a suitable time step.

In the implicit Euler method, the governing equations are given by Equation 4a and 4b.

$$\underline{y_{n+1}} = \underline{y_n} + h * f\left(\underline{y_{n+1}}\right) \tag{4a}$$

$$\underline{y_{n+1}} - h * \underline{f}\left(\underline{y_{n+1}}\right) - \underline{y_n} = 0 \tag{4b}$$

This gives us a system of non-linear equations represented as:

$$\underline{R}\left(\underline{y_{n+1}}\right) = 0 \tag{5}$$

The above non-linear system of equations is then solved using Newton's method:

$$\underline{\underline{J}}\,\underline{\partial y} = -\underline{R} \tag{6a}$$

where, $$\underline{\underline{J}} = \frac{\partial \underline{R}}{\partial \underline{y_{n+1}}} = \underline{\underline{I}} - \left( h * \frac{\partial \underline{f}\left(\underline{y_{n+1}}\right)}{\partial \underline{y_{n+1}}} \right) \tag{6b}$$

Therefore, if the initial condition is given, we can evaluate the solution at any instant of time by Explicit Euler and Implicit Euler methods by choosing the appropriate time step size.

## 3. Results and Discussion

### 3.1 Critical Points for System of Equations

For the first part of the project, the critical or stationary points of the system are found. Critical points are defined as the coordinates at which the condition given by Equation 7 is satisfied.

$$\frac{dx}{dt} = \frac{dy}{dt} = 0 \tag{7}$$

Taking this into account, we get:

$$(a - by)x - px^2 = 0 \tag{8a}$$

$$(cx - d)y - qy^2 = 0 \tag{8b}$$

This gives us 4 set of solutions which are:

$$x = 0, y = 0 \tag{9a}$$

$$x = \frac{qa + bd}{pq + bc}, y = \frac{ac - dp}{bc + qp} \tag{9b}$$

$$x = 0, y = \frac{-d}{q} \tag{10c}$$

$$x = \frac{q}{p}, y = 0 \tag{10d}$$

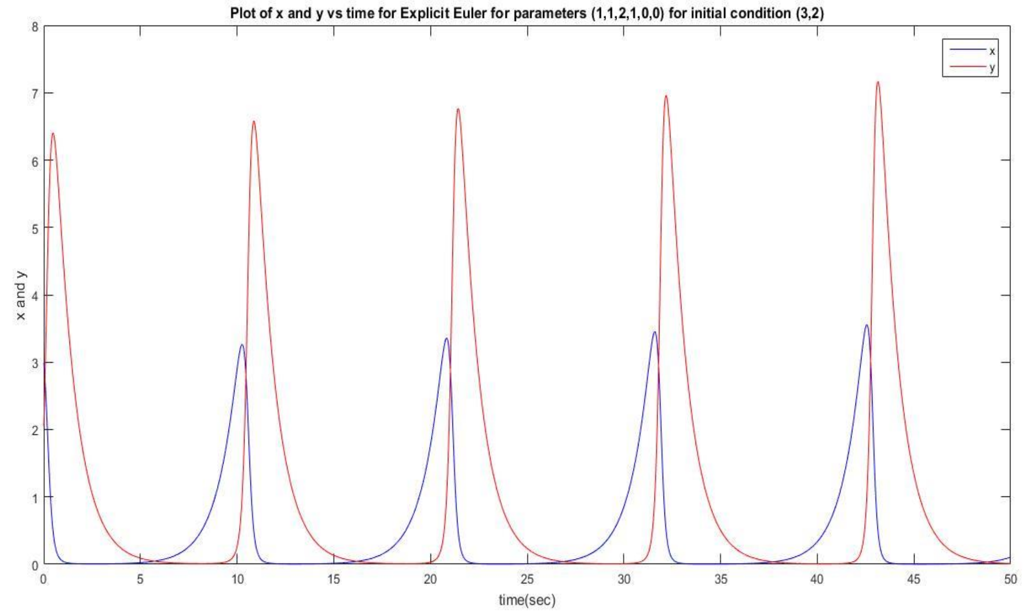
Now, if all the constants are positive, the value of the coordinates of the critical points should also be positive as populations cannot be negative. Thus, in this particular case for the system of equations, the point represented by Equation 12c is not possible. Moreover, the point represented by Equation 12b satisfies only for a given range of value of constants.

If $ac > dp$, then the non-zero physically possible stationary point is: $\left(\frac{bd + aq}{bc + pq}, \frac{ac - dp}{bc + pq}\right)$

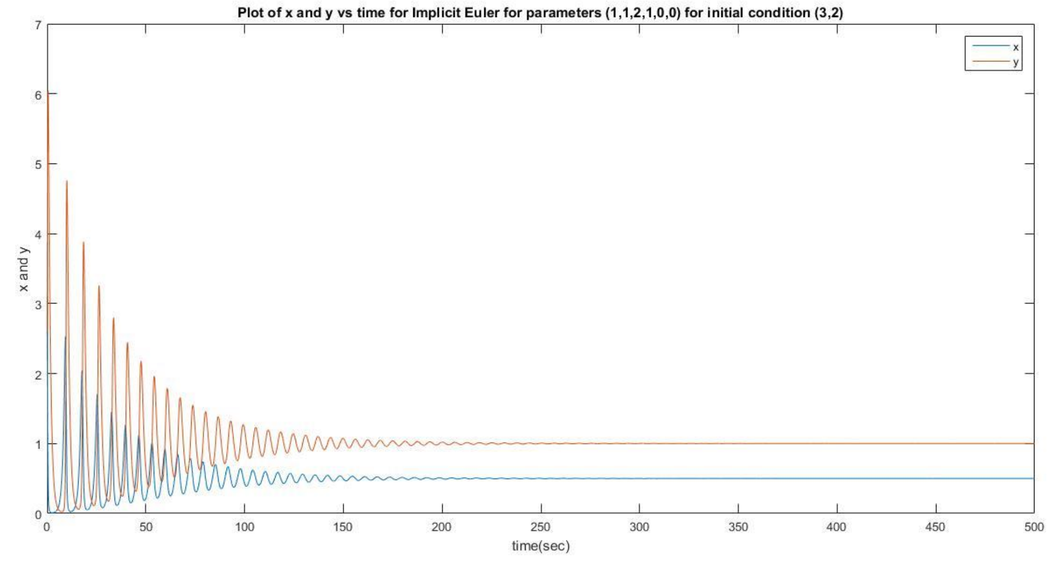
If $ac < dp$, then the non-zero stationary point is: $\left(\frac{a}{p}, 0\right)$

### 3.2 Numerical Schemes Based on Explicit and Implicit Euler Methods

The Implicit Euler and Explicit Euler methods were coded using MATLAB as a function of the parameters, time step size, the number of time steps and the initial condition. Phase plots and evolution plots were plotted for various values of the parameters and initial conditions.



(a)



(b)

Figure 1: Plot of method VS time, a) Explicit, b) Implicit

### 3.3 Accuracy Analysis of ODE Solvers

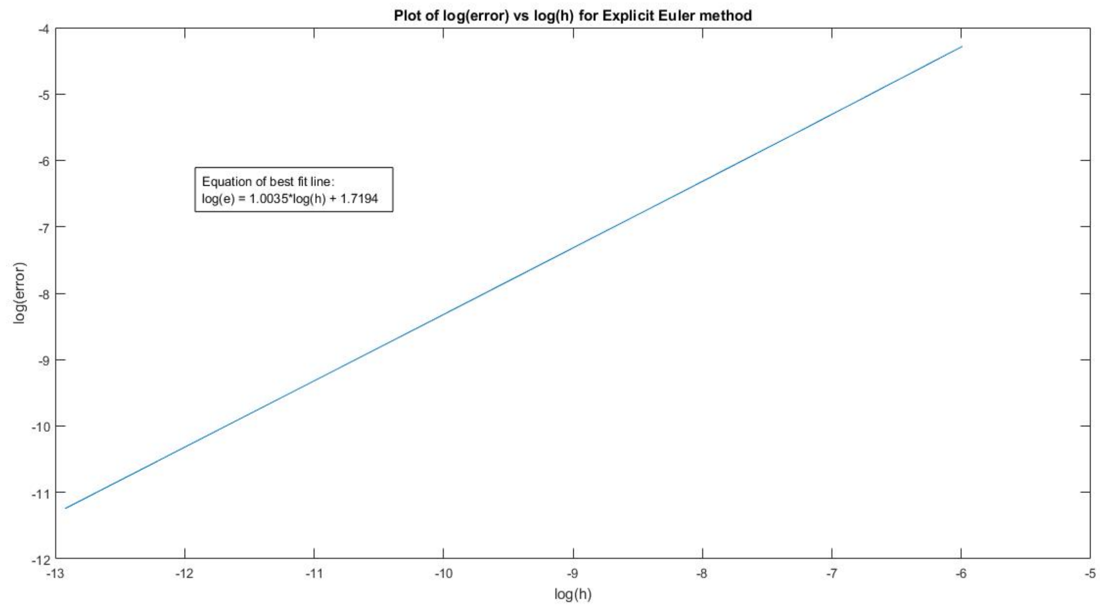
For the parameter set ( $a = b = c = d = 1$ and $p = q = 0$ ), accuracy analysis was performed for both the ODE solvers by varying the time step size. For this, initially a time step size of 0.01 and 1000 time steps were considered. The time step size was then further reduced by multiples of 10. For various time steps, the solution was then evaluated at every instant of time as in the initial case. Error was calculated between two successive solutions after reducing the time step size.

The measure of the error used in this case was the L2 norm of the error, i.e, the RMS (root mean square) of the error. The L2 norm gives an accurate representation of the difference in the solutions for two different time step sizes at all instants of time.

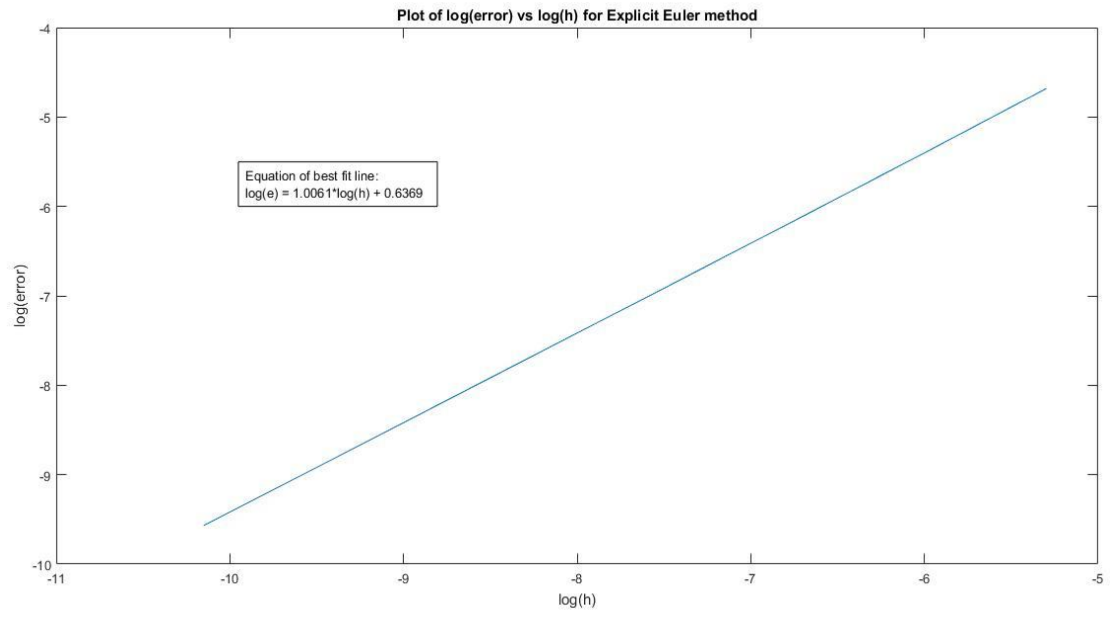
For the Implicit Euler method, the L2 norm of the difference between the solutions at time step size of $1\times10^{-6}$ and $2\times10^{-6}$ was $2.1065\times10^{-6}$. For the purposes of this project, this error is sufficiently small and the solution obtained at time step size $1\times10^{-6}$ can be assumed as the exact solution for the given system. In all further calculations, this solution is considered as the exact solution and all the errors are calculated with respect to this solution.

For the Explicit Euler method, the same cases were run and the L2 norm of the difference between the solutions was $1.31\times10^{-6}$. This error was sufficiently small for the requirements of the project and hereafter, the solution obtained at time step size of $1\times10^{-6}$ is considered as exact solution.

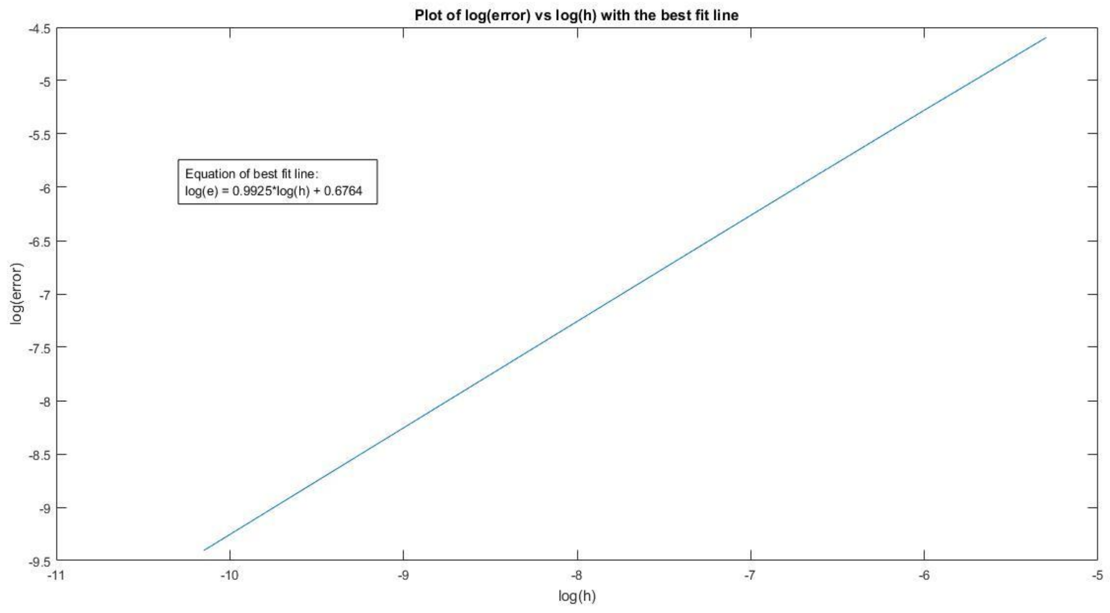
Considering the exact solution as obtained above, the error in solution at various time step sizes was calculated for both Implicit Euler and Explicit Euler methods and the logarithm of this error was plotted against the logarithm of time step size. The results are plotted below:



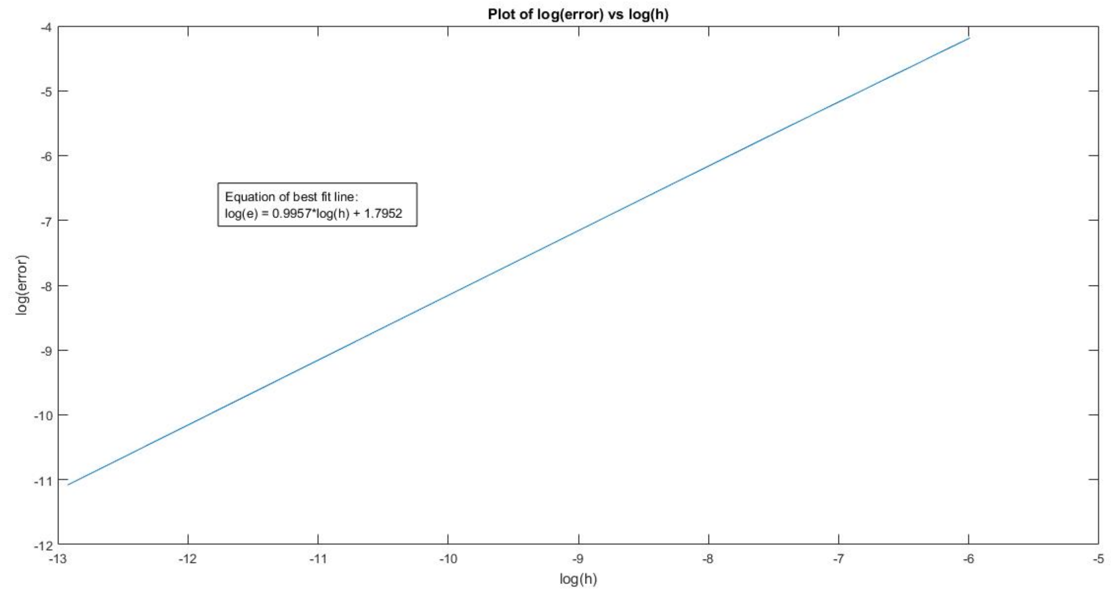
(a)


Plot of log(error) vs log(h) for Explicit Euler method
Equation of best fit line:
log(e) = 1.0061*log(h) + 0.6369
log(error)
log(h)
-4
-5
-6
-7
-8
-9
-10
-11
-10
-9
-8
-7
-6
-5

(b)


Plot of log(error) vs log(h) with the best fit line
Equation of best fit line:
log(e) = 0.9925*log(h) + 0.6764
log(error)
log(h)
-4.5
-5
-5.5
-6
-6.5
-7
-7.5
-8
-8.5
-9
-9.5
-11
-10
-9
-8
-7
-6
-5

(c)



(d)

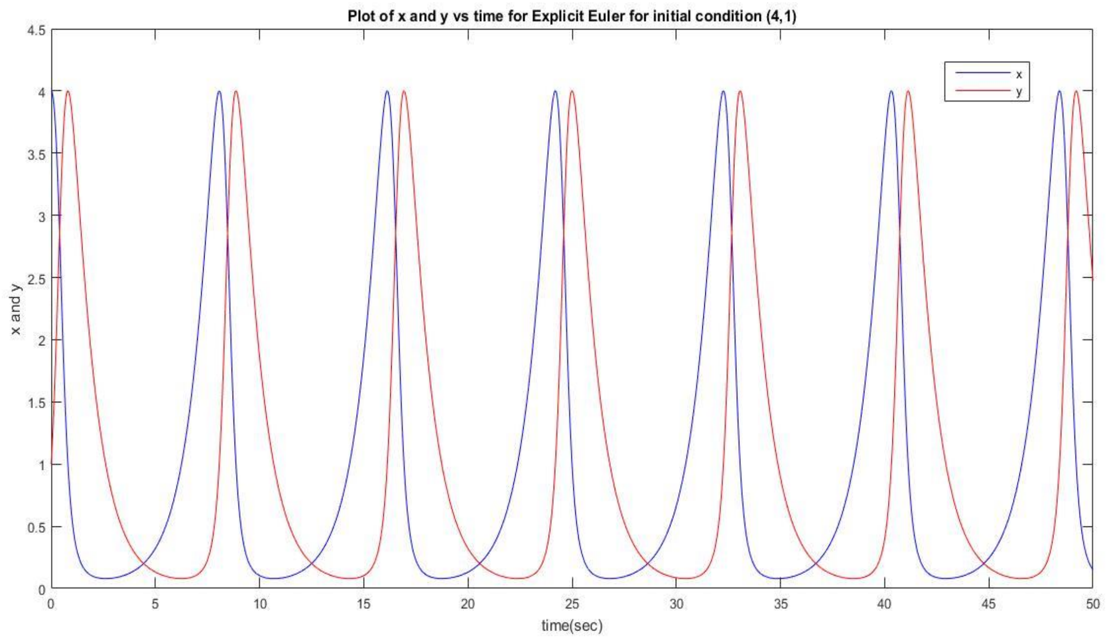
Figure 2: Slope of the line log(error) vs log(h), a) Implicit (Time Step = 1.0035), b) Implicit (Time Step = 1.0061), c) Explicit (Time Step = 0.9925), Explicit (Time Step = 0.9957)

From Figures 2a & 2b we can see that the slope of the line for two different ranges of time step size is 1.0035 and 1.0061 respectively. From this we can infer that the error obtained from the Implicit Euler method varies with time step size as approximately O(*h*). Further, from Figures 2c & 2d we can see that the slope of the line for two different ranges of time step size is 0.9925 and 0.9957 respectively. From this we can infer that the error obtained from the Implicit Euler method varies with time step size as approximately O(*h*).

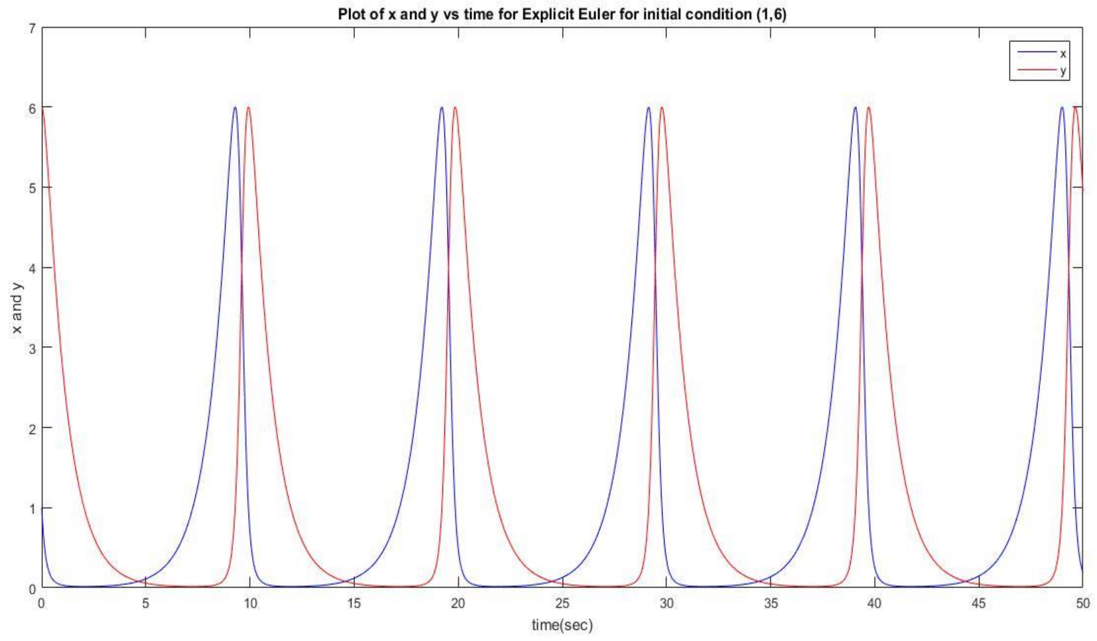
Thus we can conclude that the error for the given non-linear system of ODEs by using both the Implicit and Explicit Euler methods is approximately O(*h*). This is in line with the theoretical value of the error. The error at each time instant for both the methods is O($h^2$), and this error is evaluated over $(1/h)$ points, making the total error as O(*h*).

## 3.4 Dependence of X and Y as a Function of Time

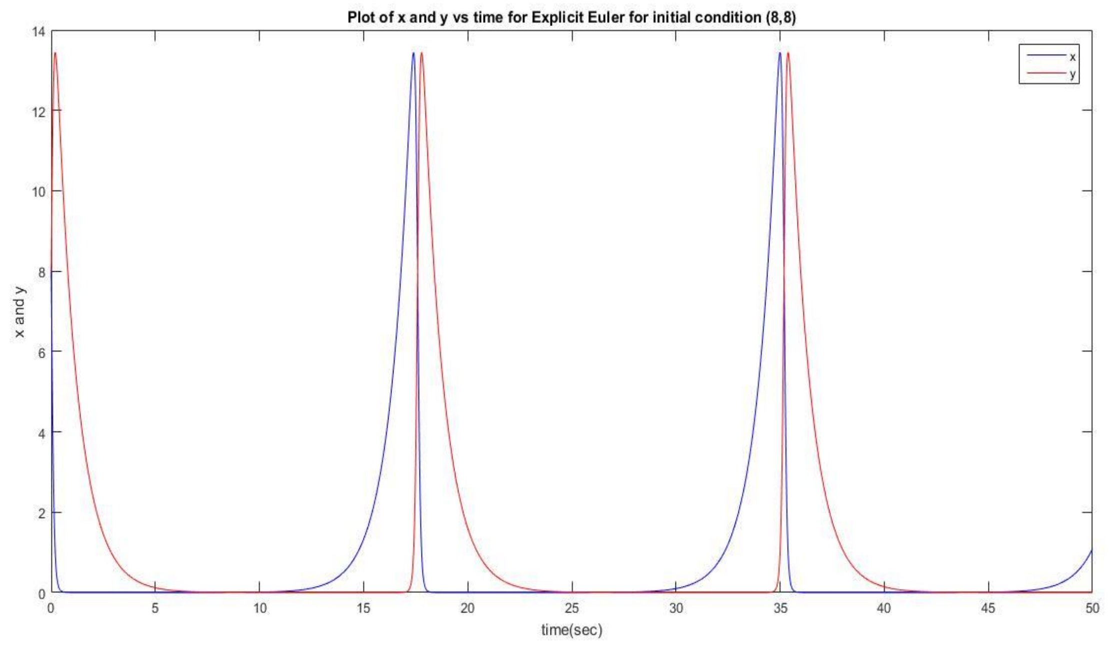
For the initial condition (1,1), it was observed that the evolution of x and y with time is just a straight line at $x = y = 1$. This implies that *x* and *y* do not evolve with time and remain at the value (1,1). This is because of the fact that at the given parameters, the value (1,1) is a critical or stationary point. Since the initial condition is already at steady state, the solution does not change with time and we get a straight line in the evolution plot. The plots for the same are shown in Figure 3.


Plot of x and y vs time for Explicit Euler for initial condition (4,1)
x
y
x and y
time(sec)

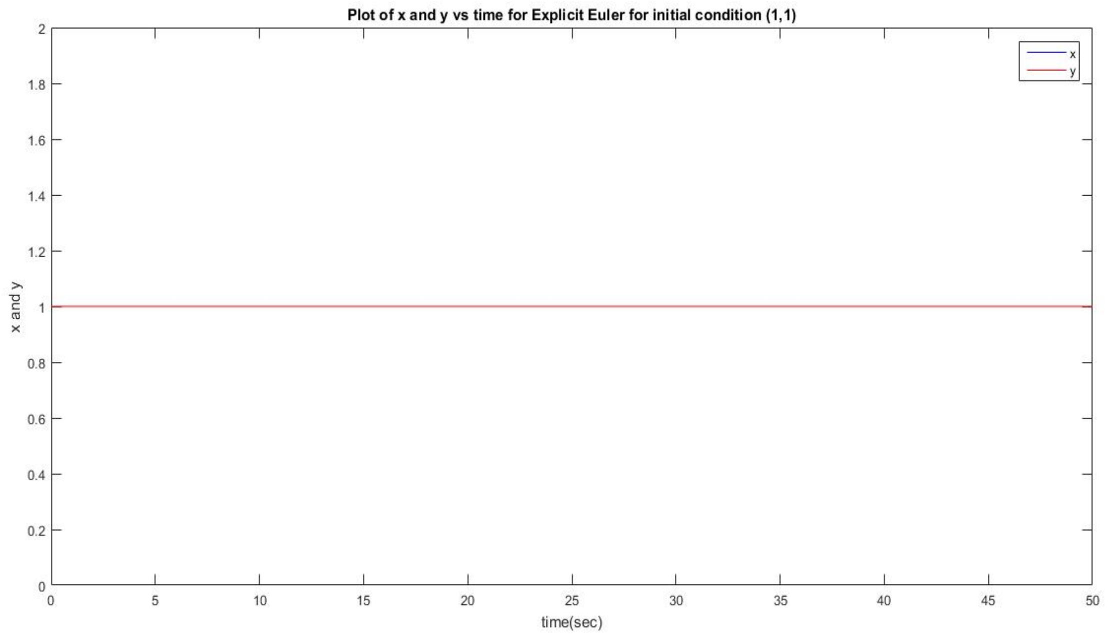
(a)


Plot of x and y vs time for Explicit Euler for initial condition (1,6)
x
y
x and y
time(sec)

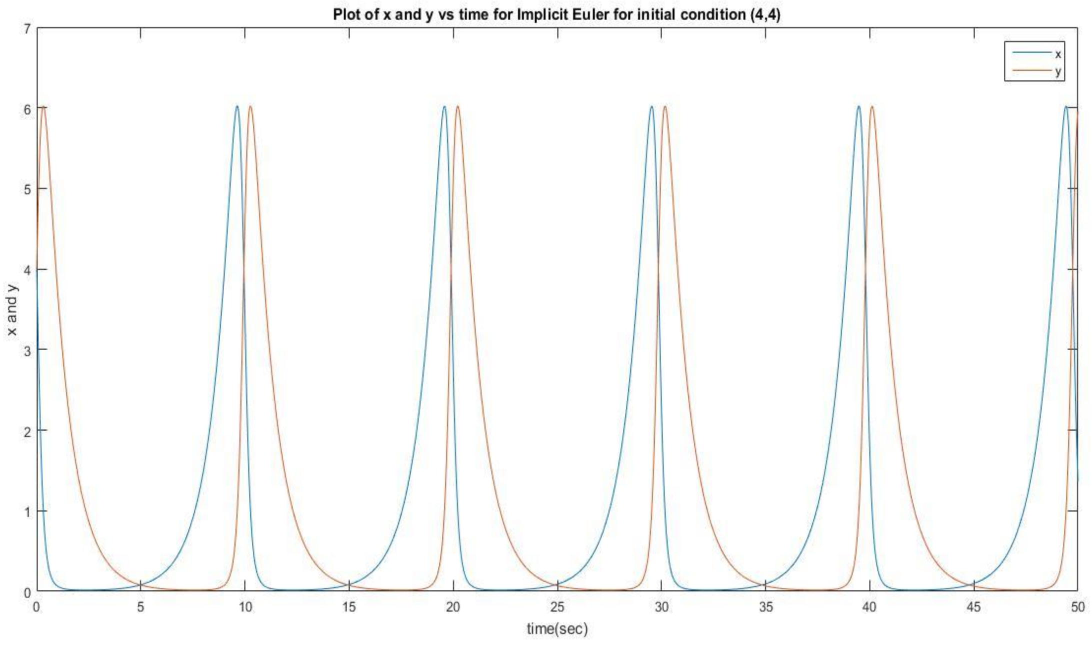
(b)


Plot of x and y vs time for Explicit Euler for initial condition (8,8)
x
y
x and y
time(sec)

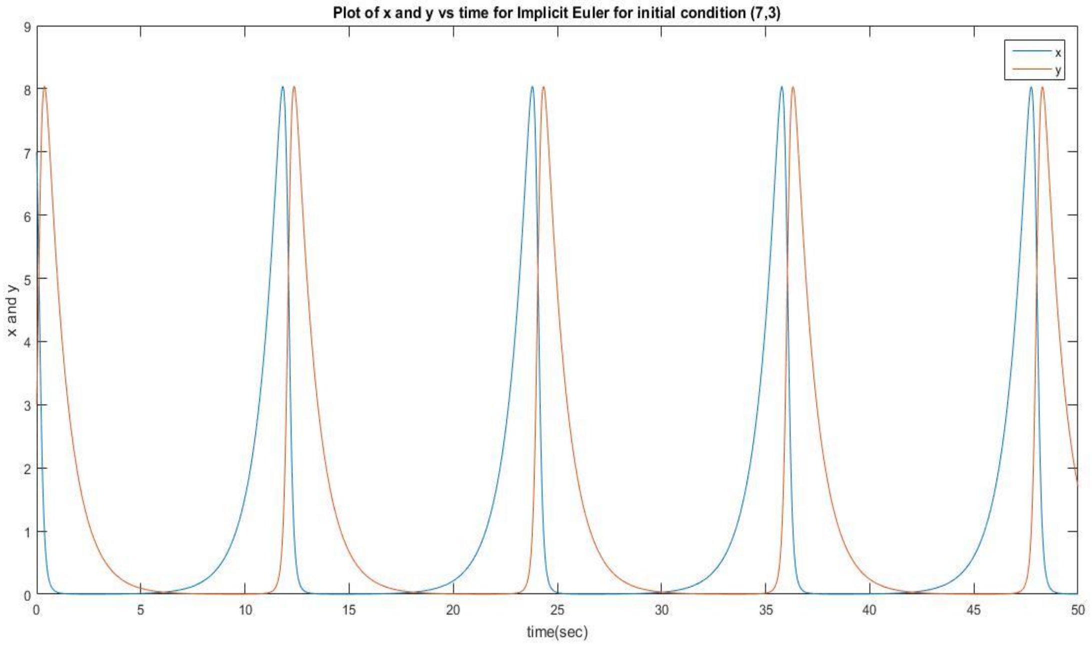
(c)


Plot of x and y vs time for Explicit Euler for initial condition (1,1)
x
y
x and y
time(sec)

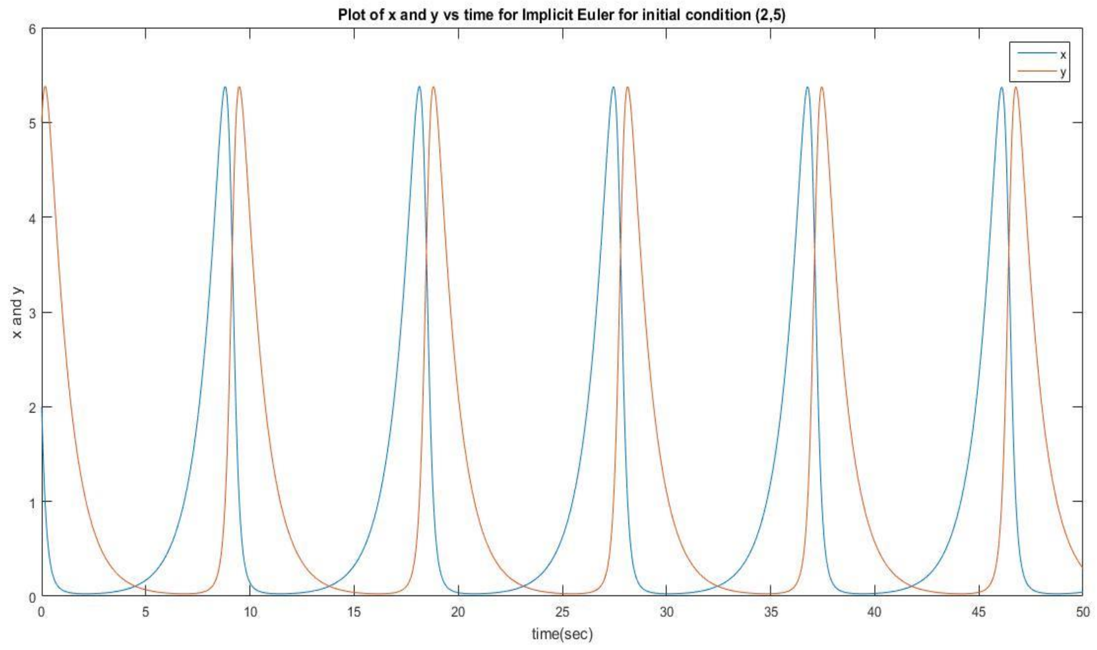
(d)


Plot of x and y vs time for Implicit Euler for initial condition (4,4)
x
y
x and y
time(sec)
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30
35
40
45
50

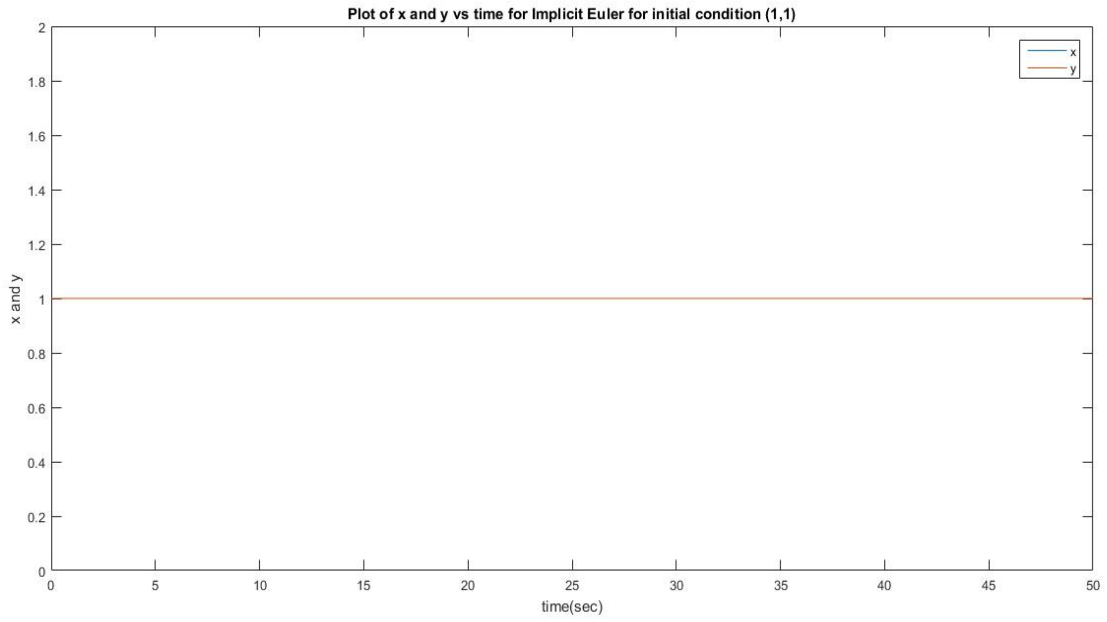
(e)


Plot of x and y vs time for Implicit Euler for initial condition (7,3)
x
y
x and y
time(sec)
0
1
2
3
4
5
6
7
8
9
0
5
10
15
20
25
30
35
40
45
50

(f)



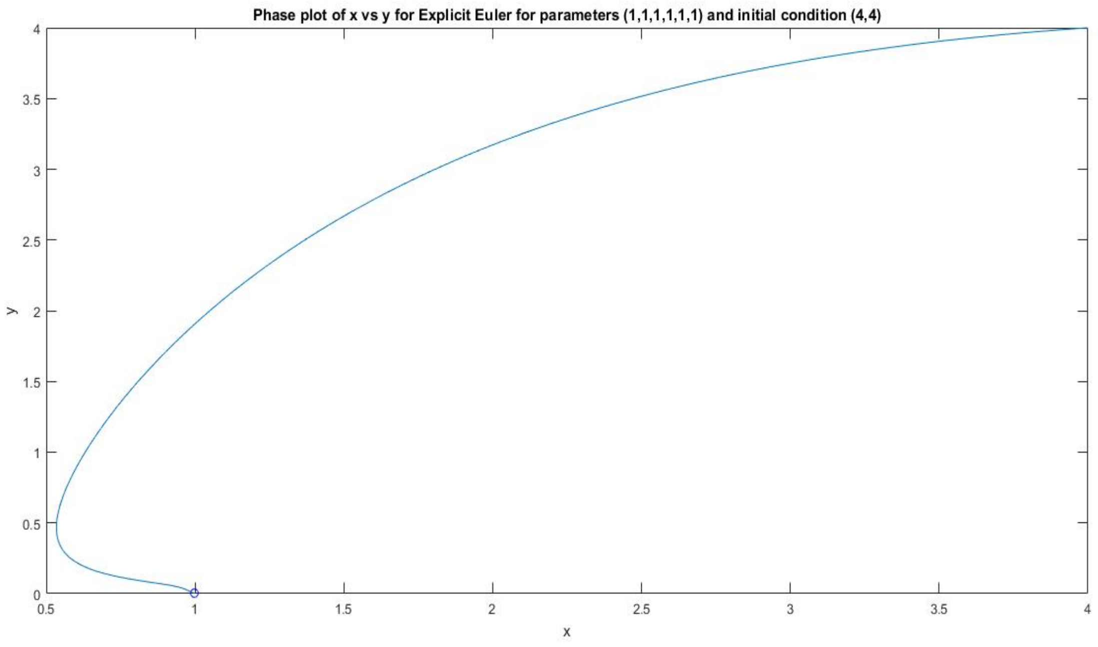
(g)



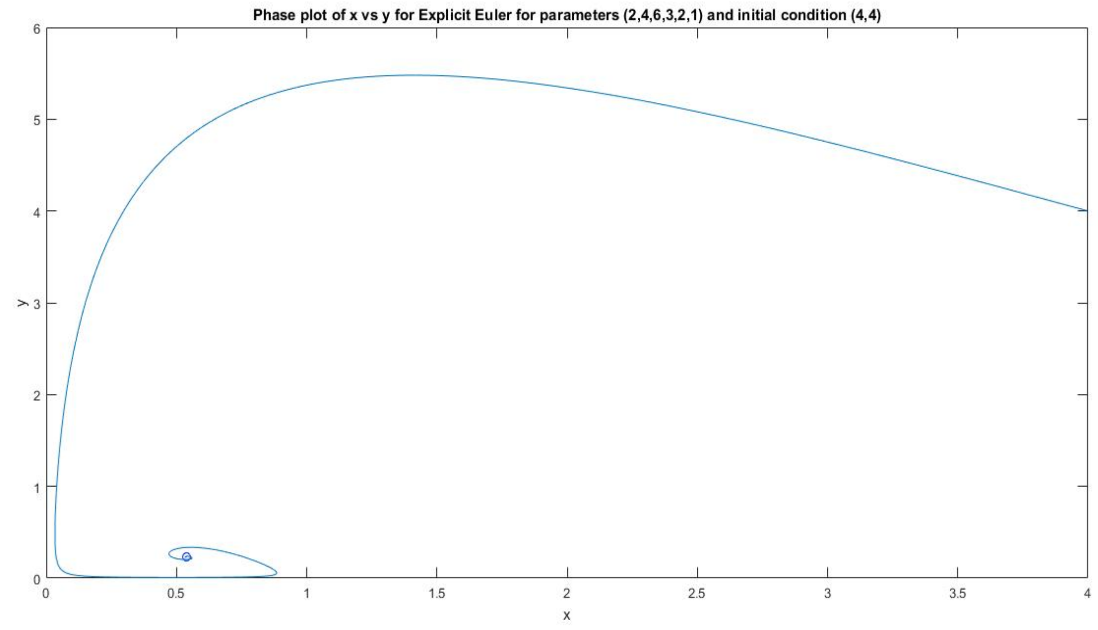
(h)

Figure 3: Plot of X and Y VS Time for different initial conditions, (a),(b),(c),(d) Explicit Euler, (e),(f),(g),(h) Implicit Euler

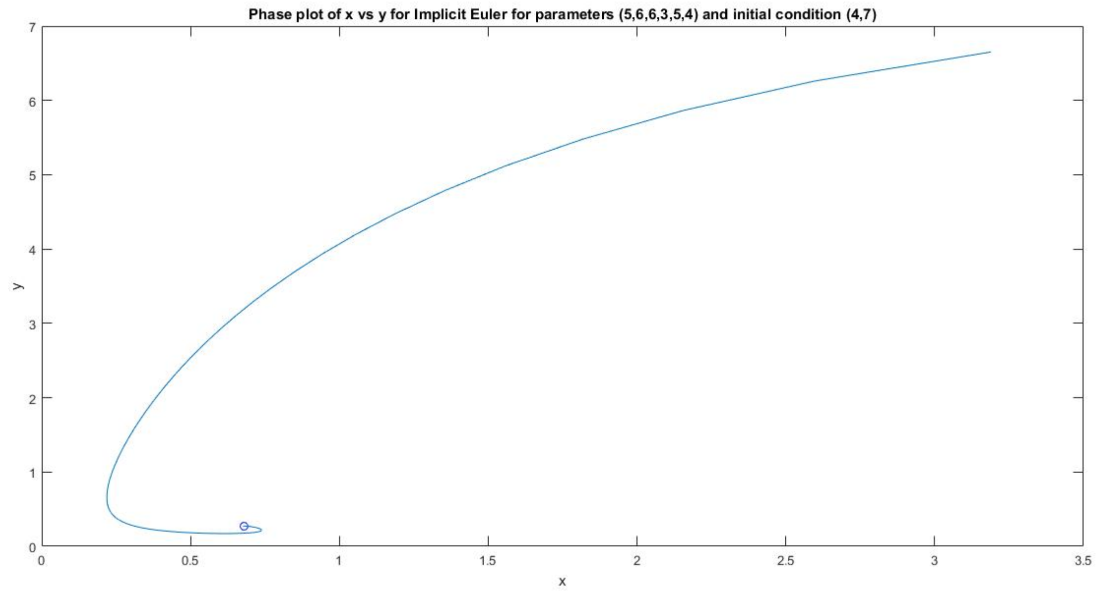
## 3.5 Phase Plots



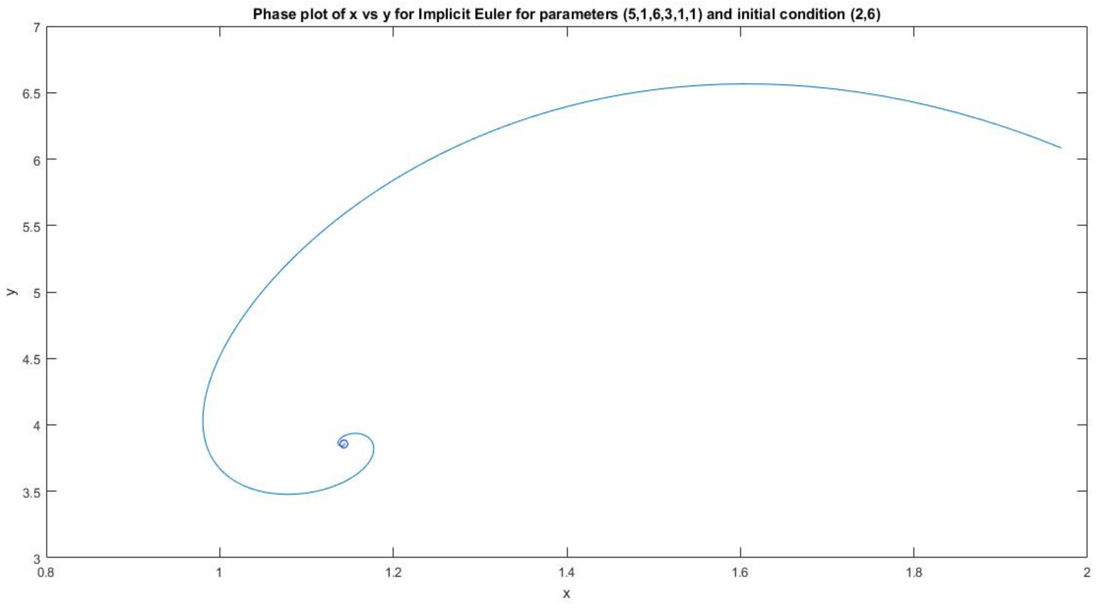
(a)



(b)



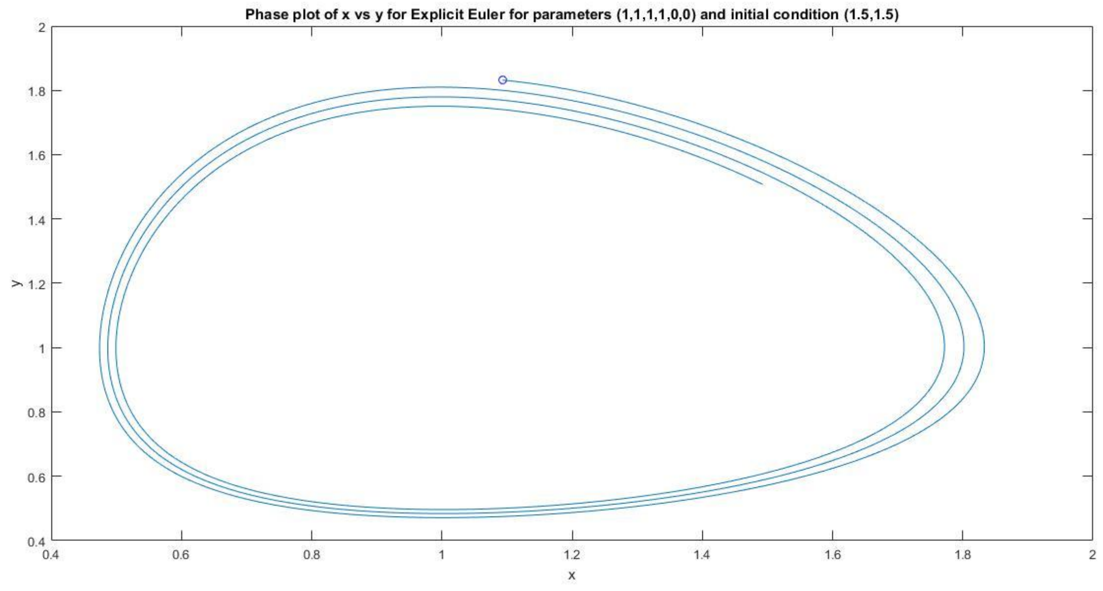
(c)



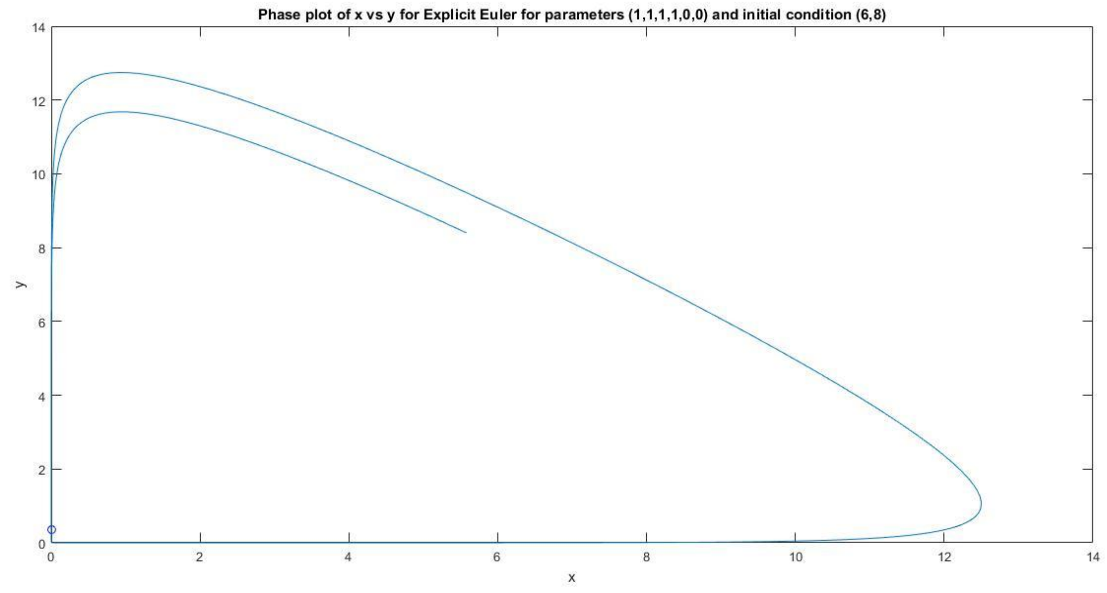
(d)

Figure 4: Phase plots of methods for different parameter values and initial conditions, a),b) Explicit Euler, c),d) Implicit Euler

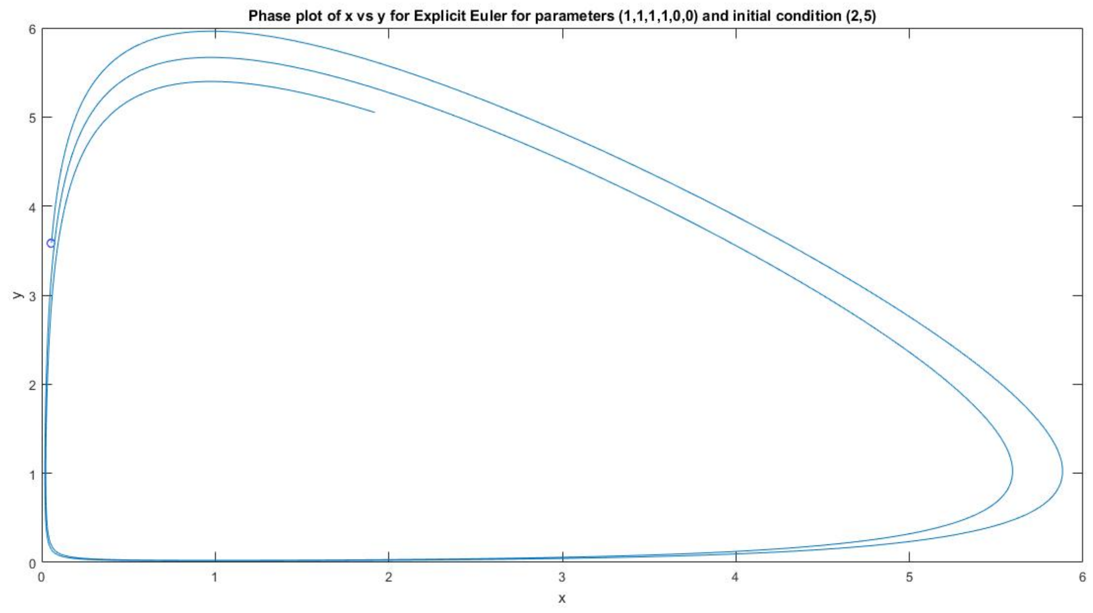
As can be seen from the above figures, the phase plots converge towards the steady state solution, i.e, the critical or stationary points of the system. Different initial values just take different number of time steps to converge but as long as the system is stable, they will converge to the stationary points. Further, we change the values of the parameters as $a = b = c = d = 1$ and $p = q = 0$.


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (1.5,1.5)
y
x
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

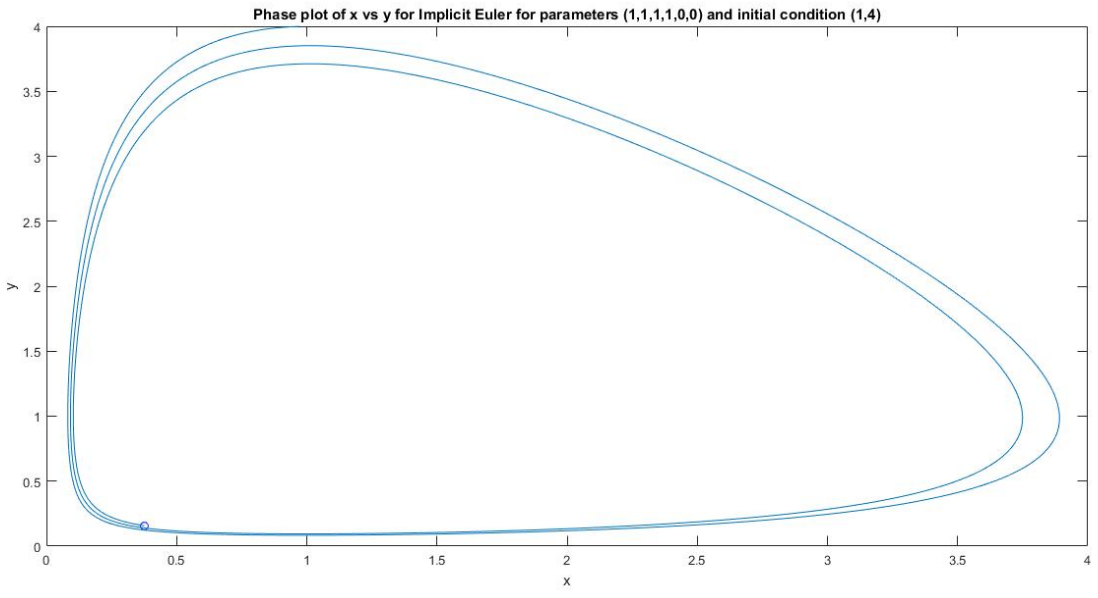
(a)


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (6,8)
y
x
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14

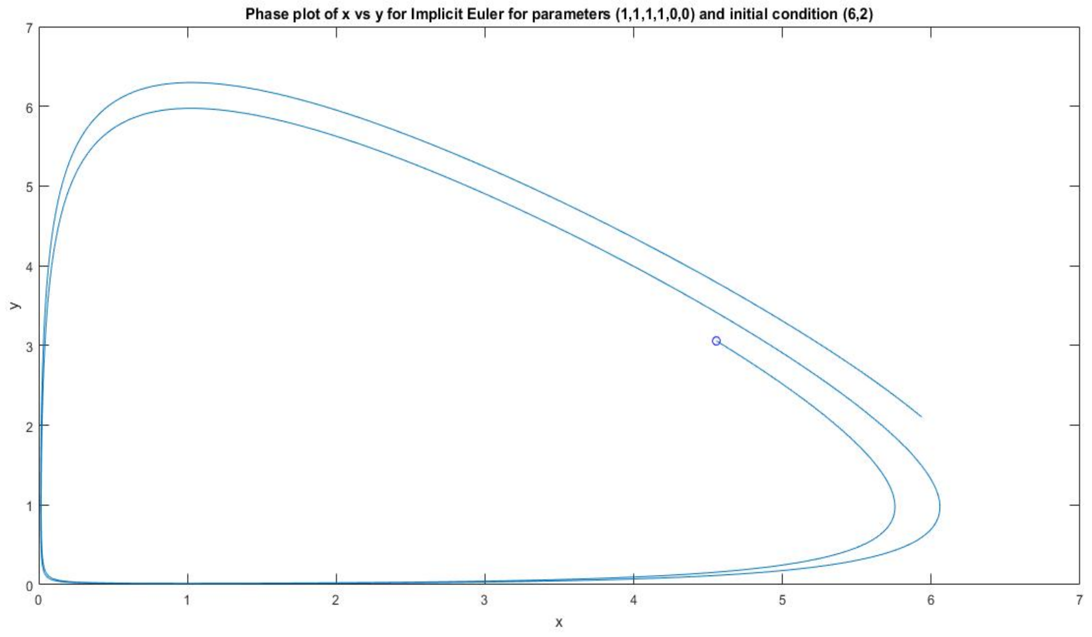
(b)


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (2,5)
x
y
0
1
2
3
4
5
6

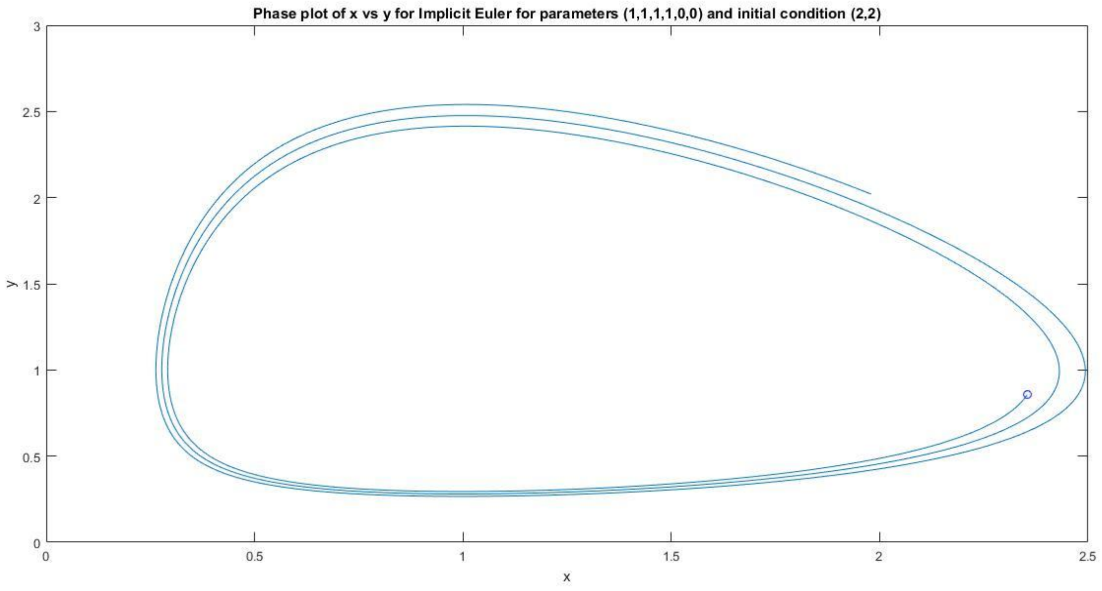
(c)


Phase plot of x vs y for Implicit Euler for parameters (1,1,1,1,0,0) and initial condition (1,4)
x
y
0
0.5
1
1.5
2
2.5
3
3.5
4

(d)



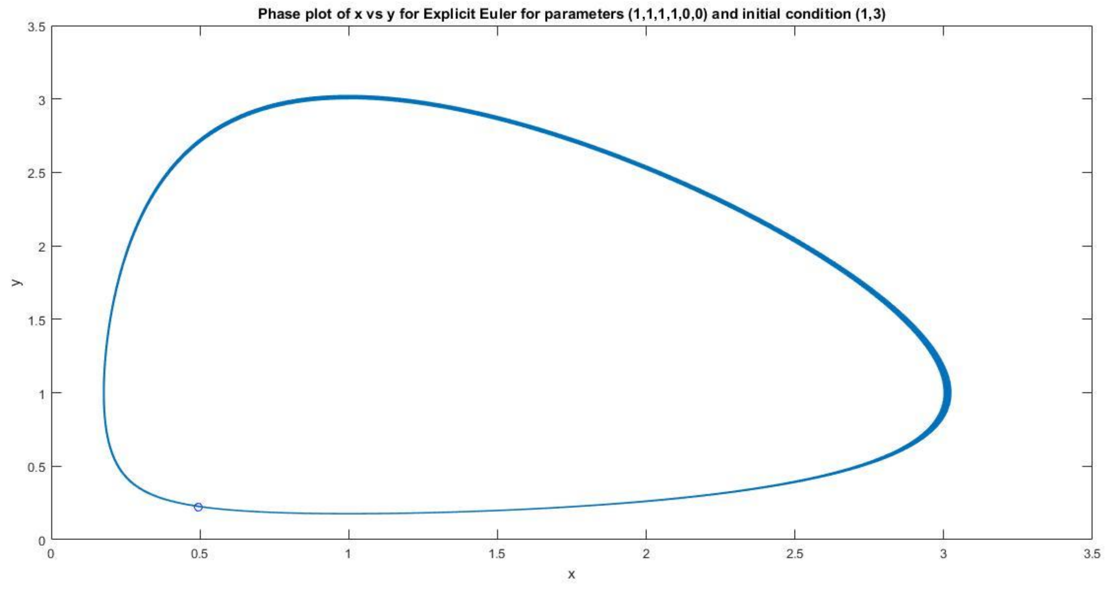
(e)



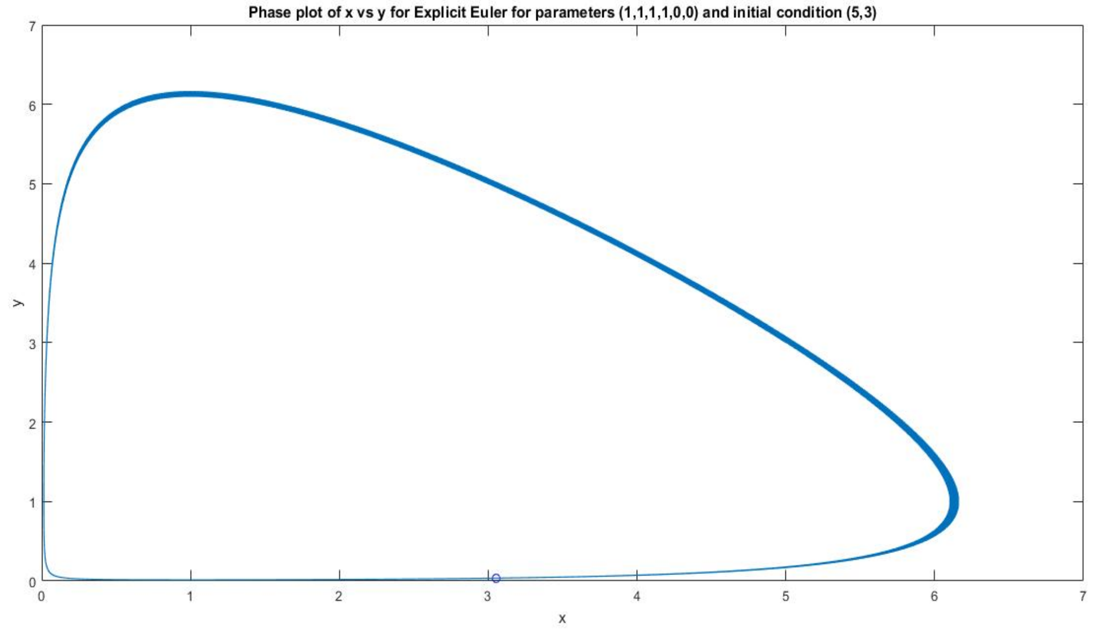
(f)

Figure 5: Phase plots for Euler methods for different initial conditions for set parameter values, a),b) Explicit, c),d) Implicit

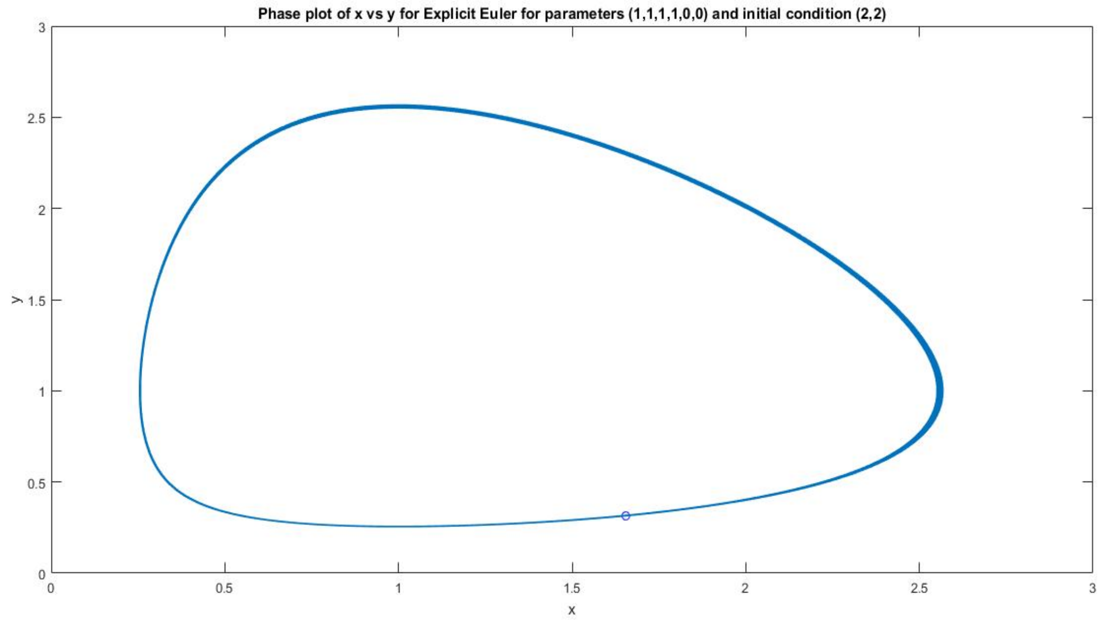
For the same set of values of parameters as above, we solve the system of equations for a very large number of time steps. After doing so, we plot the findings shown in Figure 6. As we start increasing the time steps, it is observed that the solution continues to spiral inwards, i.e, converge to the critical point.


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (1,3)
y
x
0
0.5
1
1.5
2
2.5
3
3.5

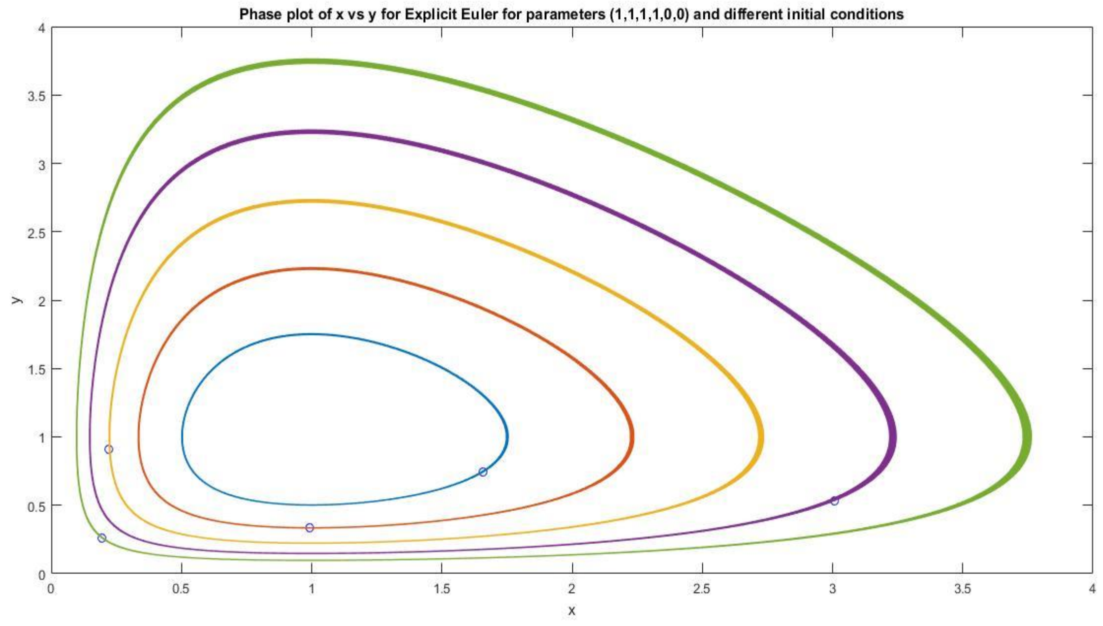
(a)


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (5,3)
y
x
0
1
2
3
4
5
6
7

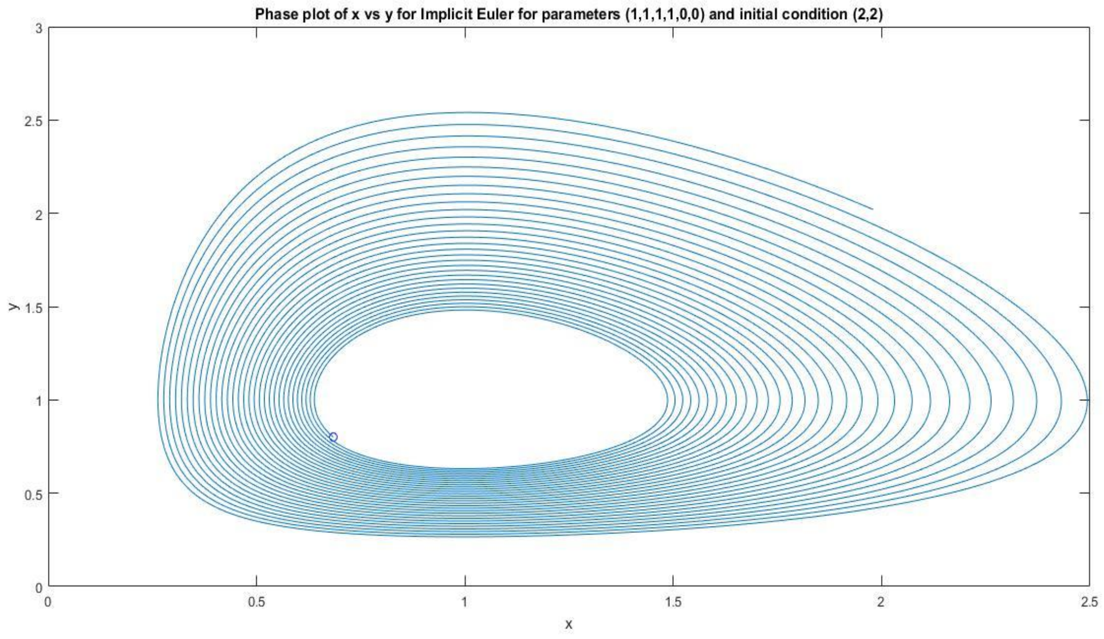
(b)


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and initial condition (2,2)
x
y
0
0.5
1
1.5
2
2.5
3

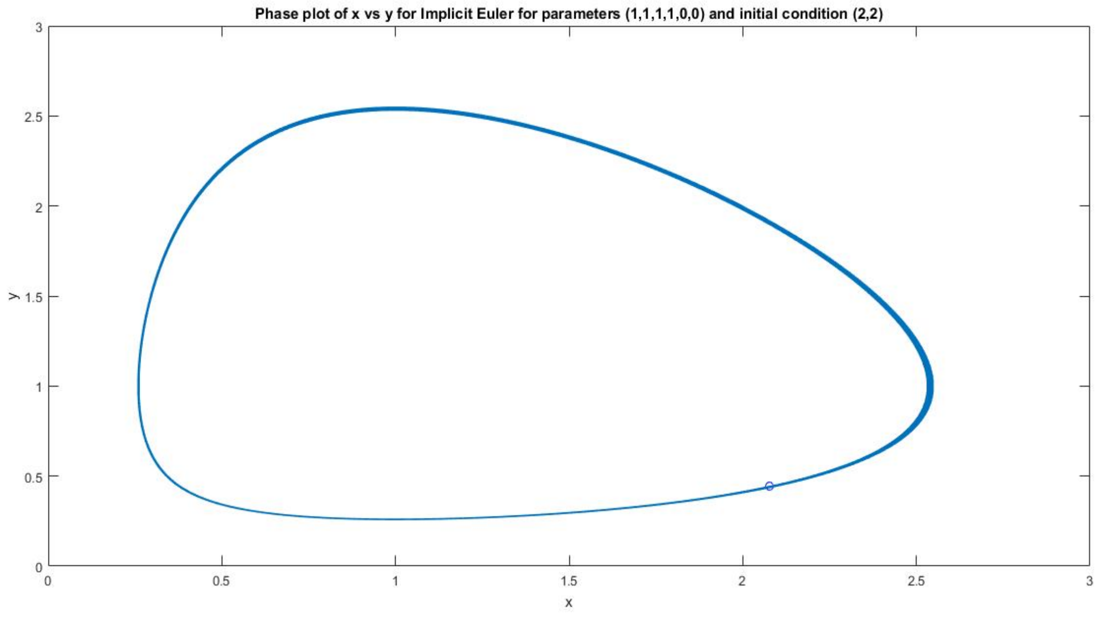
(c)


Phase plot of x vs y for Explicit Euler for parameters (1,1,1,1,0,0) and different initial conditions
x
y
0
0.5
1
1.5
2
2.5
3
3.5
4

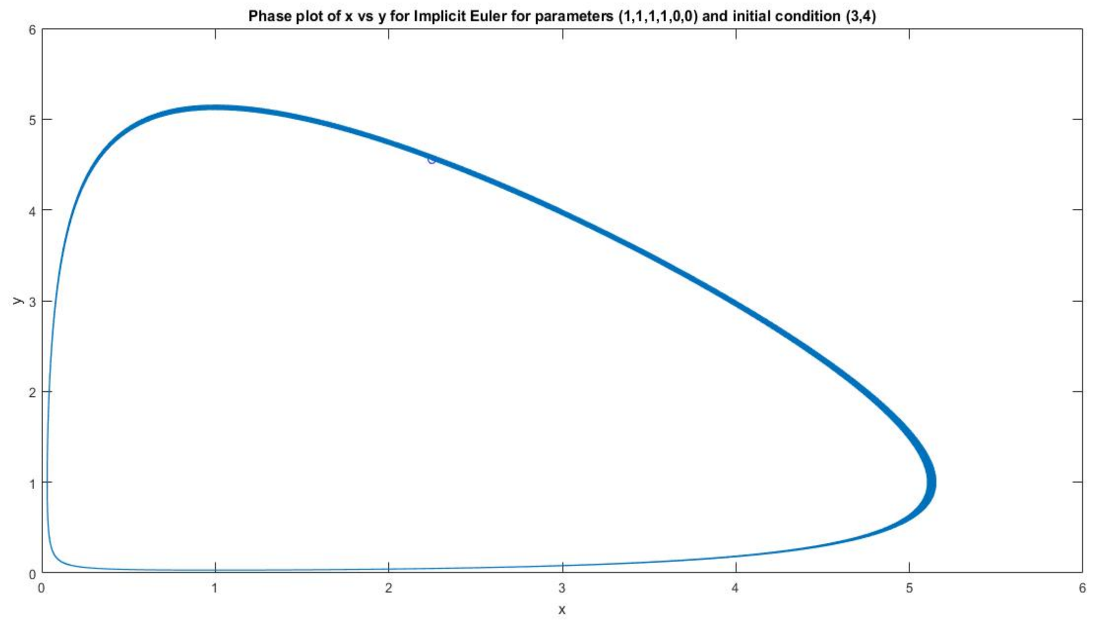
(d)


Phase plot of x vs y for Implicit Euler for parameters (1,1,1,1,0,0) and initial condition (2,2)
y
x
0
0.5
1
1.5
2
2.5
3

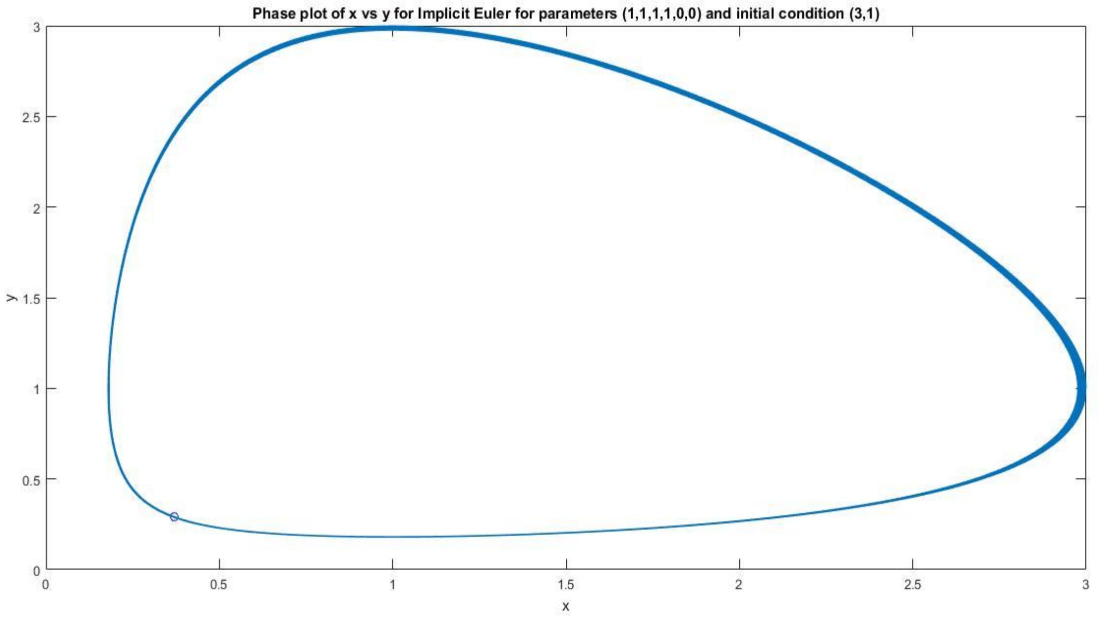
(e)


Phase plot of x vs y for Implicit Euler for parameters (1,1,1,1,0,0) and initial condition (2,2)
y
x
0
0.5
1
1.5
2
2.5
3

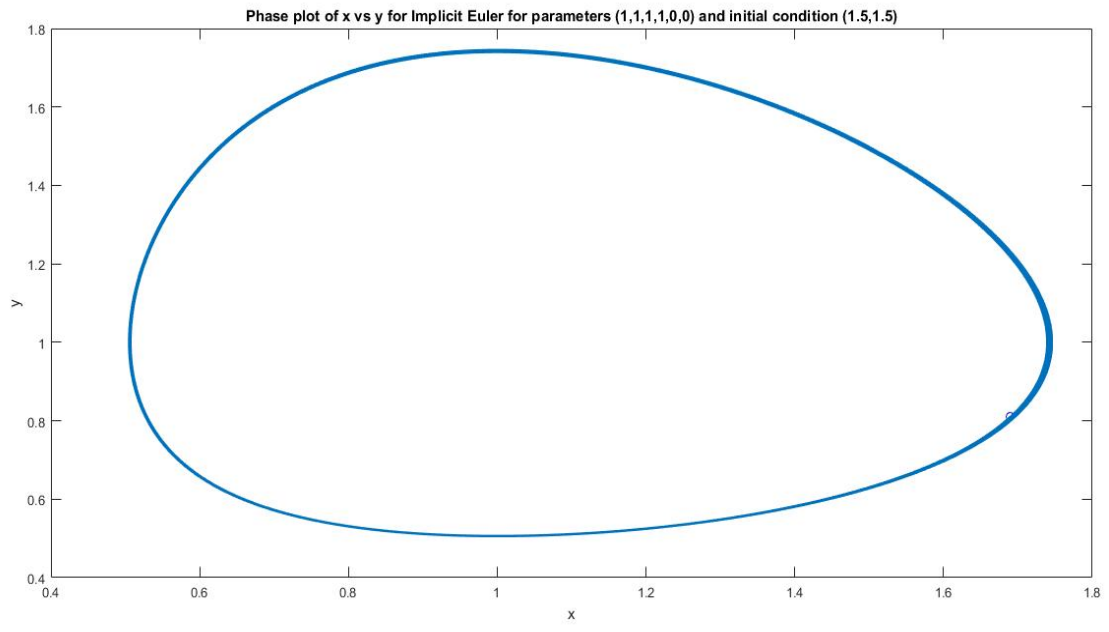
(f)


Phase plot of x vs y for Implicit Euler for parameters (1,1,1,1,0,0) and initial condition (3,4)
y
x
0
1
2
3
4
5
6
0
1
2
3
4
5
6

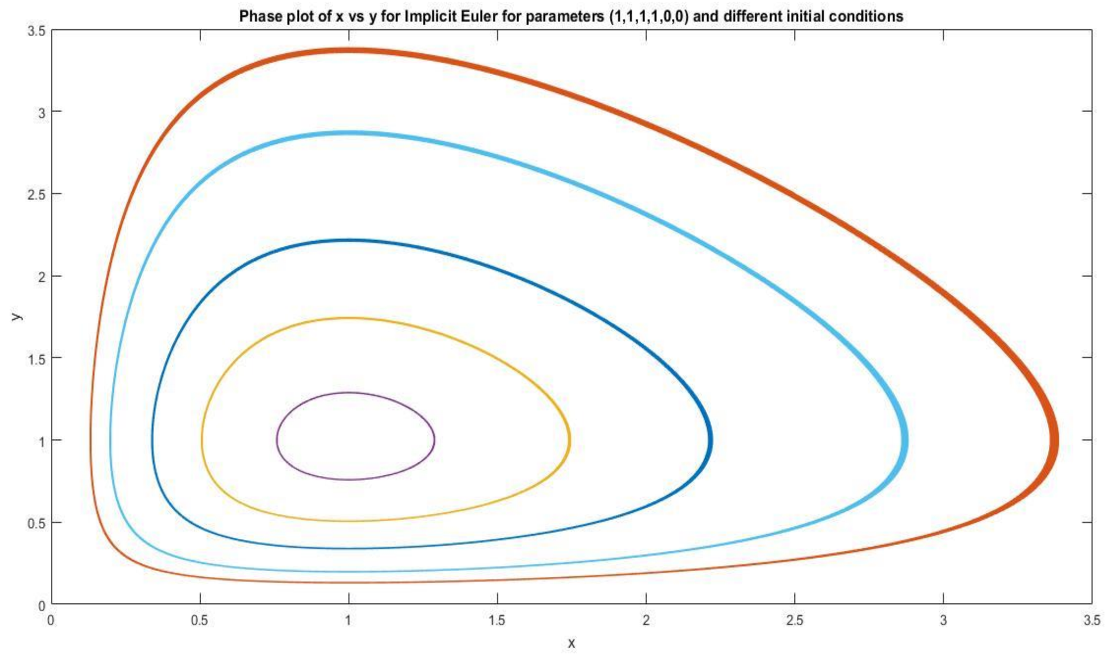
(g)


Phase plot of x vs y for Implicit Euler for parameters (1,1,1,1,0,0) and initial condition (3,1)
y
x
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2
2.5
3

(h)



(i)



(j)

Figure 6: Phase plots for different Euler methods for given parameter set, a),b),c),d) Explicit, e),f),g),h),i),j) Implicit

In this particular system, the predators survive when there are plenty of prey. As a result the population of preys decreases and thus there is no food for the predator which leads to decline in their population. At the same time, the population of the preys will increase due to natural reproduction and also due to the decline of predators. When the prey population increases, the population of predators again starts increasing due to availability of food. This chain continues and this is what is represented in the phase plots when integrated over a long time. It will take an extremely large amount of time for the system to reach some steady state as this cycle keeps on repeating. If we start with initial value of (1,1), then we do not get any such loops since (1,1) is a steady state for the set of parameters used. Any deviation from this steady state value will cause this loops to occur.

## 3.6 Numerical Stability of Explicit Euler Method

For any given set of parameters, the critical or stationary points were found and the Jacobian matrix was evaluated at this particular point. The Jacobian matrix for the stability analysis is as given below. Linear stability analysis was performed for the Explicit Euler method. In this analysis we expand $\underline{f}\left(\underline{y}_0+\underline{\delta}(t)\right)$ using Taylor series expansion and keeping only the linear term involving the perturbation. This kind of analysis means linearizing the perturbation from the steady state. Also since Taylor series expansion was used the assumption is that the perturbation is extremely small.

$$\frac{\partial \underline{y}}{\partial t}=\underline{f}\left(\underline{y}\right) \tag{11a}$$

$$\underline{y}(t)=\underline{y}_0+\underline{\delta}(t) \tag{11b}$$

$$\frac{d\underline{\delta}}{dt}=\underline{\underline{J}}\,\underline{\delta} \tag{11c}$$

$$J_{ij}=\frac{\partial f_i}{\partial y_j} \tag{11d}$$

As per above equations, the Jacobian is given by:

$$\underline{\underline{J}}=\begin{pmatrix} a-by-2px & -bx \\ cy & cx-d-2qy \end{pmatrix} \tag{12}$$

$\underline{f}\left(\underline{y}_0\right)=0$, where $\underline{y}_0$ is the steady state point.

A MATLAB code was written to calculate the critical or stationary points for any given parameter set and also to calculate the Jacobian at that particular critical point. The eigenvalues of the Jacobian were calculated. For the system to be physically stable, the real parts of all the eigenvalues should be negative. Even if one of the eigenvalues is positive, the system is said to be unstable.

For calculating numerical stability, the critical point was found for the given parameter set and the Jacobian was evaluated at that point. If real parts of all the eigenvalues were found positive, an initial value was taken very close to the critical point and convergence was checked for the Explicit Euler method for this initial condition and various values of time step size ‘h’.

It was observed that for the perturbation of 0.01 from the critical point, the system was stable for the condition $0\leq\lambda_{\max}h\leq1.86$. As the perturbation was further reduced, the stability criterion approached $0\leq\lambda_{\max}h\leq2$. This agrees with the theoretical stability criterion for numerical stability. Hence, if the perturbation is made infinitesimally small, the numerical stability criterion is

$0 \le \lambda_{\max} h \le 2$. Thus there is a restriction on the time step size that can be chosen for the Explicit Euler method.

The above condition is true only when the initial condition is very close to the critical point. If we choose the starting point further away from the critical point, the value of ‘h’ required for convergence is much less than the value obtained from the above condition. Also, the farther we start from the critical point, the value of time step size required for the system to converge decreases.

This is in contrast to the Implicit Euler method which is unconditionally stable for any value of ‘*h*’. Also, the stiffness of the system is given by

$$K = \frac{\lambda_{\max}}{\lambda_{\min}}$$

where, $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of the Jacobian respectively. For a system with high stiffness, it was observed that for the Explicit Euler method to converge, an extremely small value of time step size was required. For the parameter set ($a$=10, $b$=8, $c$=2, $d$=2, $p$=20, $q$=3), the stiffness of the matrix turns out to be 10. For this parameter set, a value of $h$ = 0.0005 is required for the Explicit Euler method to converge. If a value greater than this is taken, the solution blows up with time.

Hence, it was concluded that Explicit Euler method is convenient to use when the system is not stiff. For a stiff system, it is better to use the Implicit Euler method as we can choose any value of ‘*h*’ for stability.

## 4. Conclusion

The given set of simultaneous non linear differential equations can be solved numerically using the Explicit and Implicit Euler methods. Four critical points can be ascribed to these equations, they are: (0,0), $\left(\frac{aq+bd}{pq+bc},\frac{ac-dp}{pq+bc}\right)$, $\left(\frac{a}{p},0\right)$, and $\left(0,\frac{-d}{q}\right)$. Out of the four points only the first three are physically achievable. The solutions to the linearized equations are a pair of periodic simple harmonic functions that are phase shifted by 90 degrees. The population of prey always leads the predator population in the cycle.

The phase plots for the explicit method diverge spirally outward and the phase plots of the implicit method converge spirally inward. It can be seen that at times the phase plots imply that a high predator population can exist at a time of little or no prey population, and that this situation is always recoverable. This is not physically possible. We generally observe the model system where the predators thrive when the prey are plentiful but thin their food supply and decline.

As the predator population dwindles the prey population will increase again. This cycle is repeated. One of the physically realizable equilibrium states is represented by a fixed point in the system at which both populations maintain their current non-zero populations. This can be seen in the solution to the non-linear equation.

## 5. Appendix

The MATLAB code for implementing the implicit and explicit Euler schemes is given as follows:

```
% Explicit Euler method

function [x,y] = explicit(a,b,c,d,p,q,h,t,xi,yi)

   y = zeros(t,1);
   x = zeros(t,1);
   z = [xi; yi];
   for i = 1:t
      z0 = z;
      z(1) = z0(1) + h*((a-b*z0(2))*z0(1) - p*(z0(1)^2));
      z(2) = z0(2) + h*((c*z0(1)-d)*z0(2) - q*(z0(2)^2));
      x(i) = z(1);
      y(i) = z(2);
   end
   plot(z(1),z(2));
   hold on;
   plot(x,y);
end

%Implicit Euler method

function [x,y] = implicit(a,b,c,d,p,q,h,t,xi,yi)

   z = [xi; yi];
   y = zeros(t,1);
   x = zeros(t,1);
   time = zeros(t,1);
   for i = 1:t
      z0 = z;
      z = newton(a,b,c,d,p,q,h,z0);
      x(i) = z(1);
      y(i) = z(2);
      time(i) = i*h;
   end
   plot(z(1),z(2));
   hold on
   plot(x,y);
   hold on
%     figure;plot(time,x);
%     hold on
%     plot(time,y);
%     hold on
end
```

```
%Newton's method

function z = newton(a,b,c,d,p,q,h,z0)

    j = zeros(2,2);
    r = zeros(2,1);
    maxiteration = 10^3;
    i = 0;
    z = z0;
    while i < maxiteration
        r(1) = z(1) - h*((a-b*z(2))*z(1) - p*(z(1)^2)) - z0(1);
        r(2) = z(2) - h*((c*z(1)-d)*z(2) - q*(z(2)^2)) - z0(2);
        j(1,1) = 1 - h*(a - b*z(2) - 2*p*z(1));
        j(1,2) = 0 - h*(-b*z(1));
        j(2,1) = 0 - h*c*z(2);
        j(2,2) = 1 - h*(c*z(1) - d - 2*q*z(2));
        dz = j\(-r);
        if max(abs(dz)) < 1e-6;
            break
        end
        z = z + dz;
        i = i + 1;
    end
end
```