Modeling Population Cycles of Lynx

Patrick Pinello

Abstract. First-order differential equations are incapable of incorporating historical effects on populations, unlike second-order equations. Using MATLAB's ode45, Euler's method, and MATLAB's diff function, John P. Clark's thesis surrounding population modelling using second-order equations is confirmed. The models are verified using the real-world data found in Clark's original paper. This is done using numerical analysis and post-processing on equations developed by Clark concerning the population cycle of Canadian lynx in the 19th century.

Introduction

Traditionally, first-order differential equations have been utilized to model the populations of animals, in this case Canadian lynx. This can be seen in the famous Lotka-Volterra equations, which "are well known in mathematical biology as models describing the growth process of different species living in an isolated environment and competing for the same resources" [1]. However, it has been argued that these first-order equations leave out valuable information when modeling these types of systems. These arguments have strong philosophical elements and are not solely quantitative, but they have undeniable effects on the methods of modeling which produce more quantitative data. It is argued that "birth and death rates depend on historical effects as well as present conditions..." and this makes higher order differential equations more interesting [2]. If there was only a need for describing immediate causes of changes in a population, first order equations would be sufficient in modeling this. "However, as these effects become more remote, derivatives of higher order become important" [2]. The second order differential equation shown in Equation 1,

$$\frac{d}{dt}N' = N'' = \frac{d}{dt} * p = q\left(N, N', E\right) \qquad (1)$$

is essential for expressing the effect of past events on present populations. Where q is the derivative of p, a function of environmental variables (E), the population level (N), and the rate at which the population level changes over time (N'). This function is equivalent to the second derivative of N, or the acceleration of the change in population. Using the Lotka-Volterra forms of the equations gives the following.

$$N1'' = (c - Dn1) * \ (An1 - N1') + \frac{1}{N1} * \ (N'1)^2 \quad (2a)$$

$$N2'' = (a - bN2) * \ (cN2 - N2') + \frac{1}{N2} * \ (N2')^2 \quad (2b)$$

Where "a", "b", "c", and "d" are constants relating to the specific population. These equations show the impact of one population (N2) on another (N1). Historical effects are reflected in N1 and N'1. For this specific system, the population of lynx across a region in Canada can provide good data for determining these equations' effectiveness in modeling population and the effects of historical events.

In an expansion of this idea, it has been questioned that "from a philosophical point of view, [one] would ask whether historical characteristics of an organism are important to that organism in any way other than as represented by its current state" [3]. The importance of historical events in determining current data is one that will be determined by the previously defined second order differential equations. The utilization of these equations requires the analysis of the "prey isocline" and the restrictions on growth functions [4]. These second order equations and methods will be applied to a set of data involving lynx populations [5].

$$Gk = a(K - N) \qquad (3)$$

Equation 3 represents the tendency for a population to be affected more strongly the farther it is away from its equilibrium position, "K". "a" is the frequency at which the population booms. N is the population.

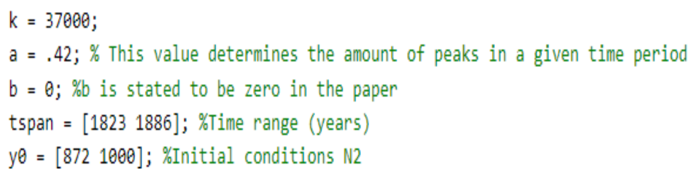
$$Gh = - b * N' \qquad (4)$$

Equation 4 represents the previous history of the population, where "b" is a constant empirically determined to be zero. It must be zero for this model to work. These combine to give the equation below.

$$N'' = a(K - N) - bN' \qquad (5)$$

Equation 5 will serve as the basis for this model. It will be transformed into a system of two first-order ordinary differential equations. This will then be solved using MATLAB's ode45 function and Euler's method. MATLAB's ode45 uses a Runge-Kutta method. It takes larger steps and performs more computations than ode23. It is ideal for ordinary differential equations with smooth solutions. It integrates the given system of equations over a certain time span given initial conditions. Euler's method uses the concept of linear approximation. Tangent lines are utilized to approximate the solution to a system of equations given their initial values.

Figure 1 shows the defining of necessary constants. K is determined empirically, or in this case by looking at the original data and seeing the spread. The middle of the population is about 37,000. "a" is the frequency. ".42" gives the exact number of peaks necessary to model this time period, "tspan". The initial conditions of N2 are empirically determined or in this case developed from looking at the original data. The population starts at 872 and 1000.



```
k = 37000;
a = .42; % This value determines the amount of peaks in a given time period
b = 0; %b is stated to be zero in the paper
tspan = [1823 1886]; %Time range (years)
y0 = [872 1000]; %Initial conditions N2
```

Figure 1 – Defining Constants

Methods

This section will discuss the various methods developed to simulate the population cycles of lynx.

Implementing ode45

The first step was to deconstruct Equation 5 into two first-order differential equations. This was done by first distributing and rearranging. This gave Equation 6.

$$N'' = -bN' - aN + ak \quad (6)$$

Equations 7, 8 and 9 further the development of this equation.

$$N1 = N \,(7)$$

$$N2 = N1' = N' \,(8)$$

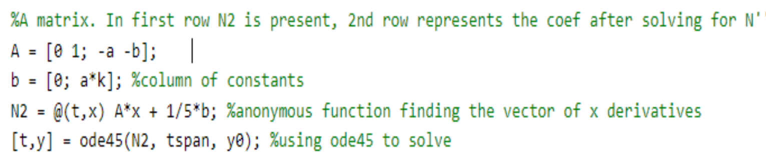
$$N3 = N2' = N'' \,(9)$$

These equations led to Equation 10 and 11.

$$N1' = N2 \,(10)$$

$$N2' =- bN2 - aN1 + aK \,(11)$$

Equation 10 and 11 now gives the system of first-order differential equations that will be used for the model. This leads to the matrix "A" to be given as [0 1; -a -b]. Matrix "b" is a column of constants. As mentioned earlier the first value must be set to 0, so it is determined to be [0; a*k]. With this, the anonymous function will be constructed. To account for the units used, "b" is multiplied by one-fifth. This anonymous function is then called using ode45. This can be seen in Figure 2.



```
%A matrix. In first row N2 is present, 2nd row represents the coef after solving for N'
A = [0 1; -a -b];
b = [0; a*k]; %column of constants
N2 = @(t,x) A*x + 1/5*b; %anonymous function finding the vector of x derivatives
[t,y] = ode45(N2, tspan, y0); %using ode45 to solve
```

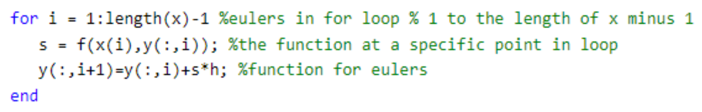
Figure 2 – Using equations with ode45

This was then plotted using MATLAB's plot function. The independent variable was defined as t, or years. The dependent variable was defined as the number of lynx, y.

Implementing Euler's Method

Utilizing the system of equations determined in the previous section, Euler's method was used to compare with ode45. First, the time span this is occurring must be defined as a single number. This was done by subtracting the final value of the previously defined "tspan" from the first value of the same vector. This was then divided by 10,000 giving the step size. The reason that 10,000 was used was because, if any number below this value was used, a solution wasn't able to be found. Due to this, Euler's method, in this instance, is more processing intensive than ode45. The total time used was then divided by the step value, this is the number of steps. With all values defined, the function to call Euler's method must be written.

It is a function with the inputs of a function, the initial independent values, the step size, the number of steps, and the initial dependent values. "x" and "y" are created as row and column vectors respectively. The first row of "x" is set to the inputted initial independent values. The first column of "y" is set to the inputted initial dependent values. A for loop is entered. To determine the number of iterations of this loop, the length function is used on the "x" vector. This is constructed by incrementing by the step value from the initial x to the initial x plus the number of steps multiplied by the step size. One is subtracted from this length. As the loop iterates, the function fills the "x" and "y" vectors with the value at that point. This is then multiplied by the step size and added to the value of "y" at that point. This is shown in Figure 3.



```
for i = 1:length(x)-1 %eulers in for loop % 1 to the length of x minus 1
   s = f(x(i),y(:,i)); %the function at a specific point in loop
   y(:,i+1)=y(:,i)+s*h; %function for eulers
end
```

Figure 3 – Euler's method with for loop

"y" as a column vector and x as a row vector are then outputted, completing the function.

This function is then called with the initial values, step size, number of steps, and function as described above. The outputted vectors are then plotted against each other. This will be compared to the plot after using ode45.

Post-Processing – Taking the Derivative

The final step of modeling this system was to perform a type of post-processing on the model. To better understand the speed at which the lynx population expands and contracts, the derivative of the model developed using ode45 was taken. The main function used in this section if MATLAB's diff function. The implementation of this can be seen in Figure 4.

```
%%% Post-Processing (Derivative) %%%
dt = diff(t); %diff of t
dy = diff(y); %Diff of x
dy_dt = dy./dt; %Derivative with element by elemnt
plot(t(1:end-1),dy_dt);
```

Figure 4 – Taking the Derivative of ode45 Model

The diff command is used to find the derivative of the years, "t", and the population, "y". These are the outputs from the original calling of ode45 on the system of equations. Then, the ratio of these values is taken. This gives the derivative of "y" in relation to "t". This is then plotted over the appropriate period of time.

All methods of modeling and processing have been completed and the results will now be shown and compared.

Results

Figure 5 below shows the model plotted using ode45 and Euler's method. The real-world data is plotted as well.

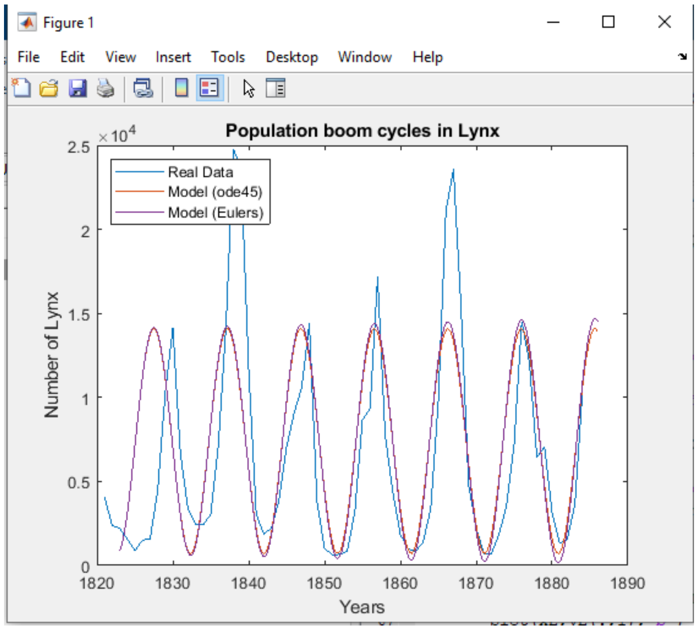


Figure 5 – Real data vs ode45 vs Euler's

Compared to the real-world data [5], the models are very constant. They don't account for outlier years such as the mid-1830s and the mid-1860s. Besides those times, however, the models are extremely accurate. They accurately model both the time it takes for a full cycle to complete (it's period), the rate at which they increase and decrease, and the maximum amplitude.

When comparing the model from using ode45 and Euler's method, it is clear they are very similar. However, there is one main difference, Euler's method provides much smoother results. When observing closely in Figure 6, it is also clear that the amplitude is slightly greater.

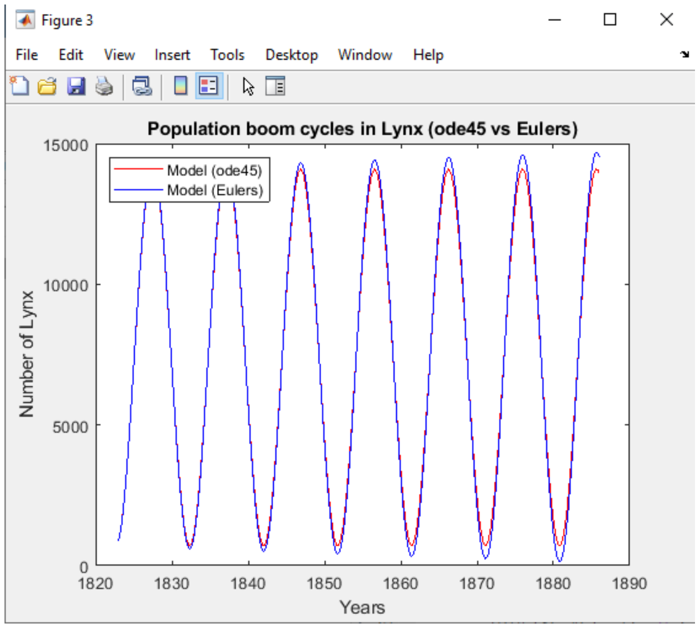


Figure 6 – Comparing ode45 to Euler's

Due to these two facts, it can be assumed that ode45 is missing some data. The reason might be because Euler's method utilizes such a small step size and can catch things that ode45 misses. As mentioned earlier, this is the smallest size it functions on. The excellent mimicking of the real-world data and the similarity between each other shows that these models have successfully modelled the system.

Next, to gain more knowledge from the system, the derivative of this model must be observed. This is the acceleration of the population of lynx. This is shown in Figure 7.

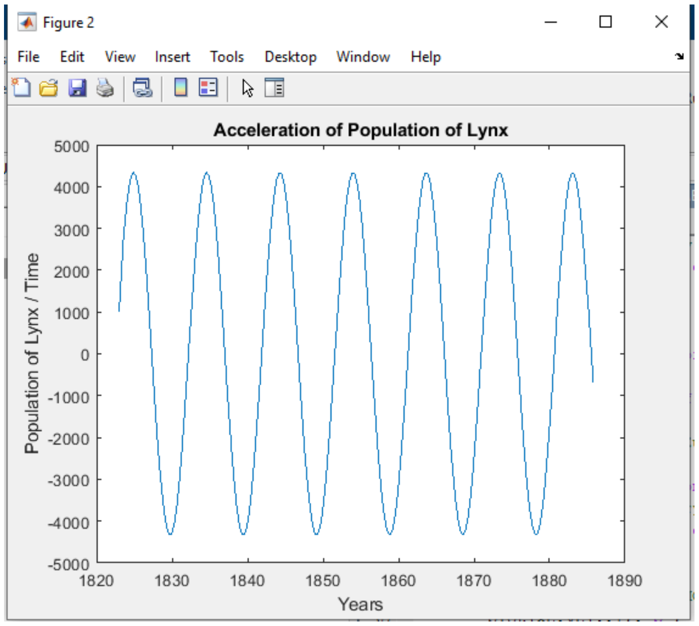


Figure 7 – Acceleration of Population of Lynx

This data, achieved using the diff function, has two main purposes. The first is to observe the trends of the population. With this information, if a conservationist is viewing the population and sees deceleration similar to the one in Figure 7, it might be cause for concern. It might allow them to make inferences about policies such as hunting the prey of the lynx or construction of temporary campgrounds. Additionally, if they see an abnormal acceleration of population it may be beneficial to investigate the cause. For example, if a team it attempting to revitalize a population of lynx, they would be attempting numerous techniques to do so. This data could act as a control to compare the new attempts against.

Also, the minima could easily be found using this data. Whenever the y value is 0 in Figure 7 and was previously accelerating, the population is at a minimum. This would be useful for keeping track of the population's development. The environmental factors leading to minima could therefore be studied more easily if the conservationist understands the trend. If the y value is 0 after a period of deceleration, the point is a maximum. This can also be useful when comparing the effects of the last year on the next. Overall, the derivative and thus the acceleration of the population can give a lot of detail to the model and to those who can use it.

Finally, it is clear that this model of the derivative is correct due to its similarity to a cosine wave. The initial wave plotting the population versus time is similar to a sine wave as it starts close to zero. After taking the derivative, this should give a cosine wave. The model of the derivative has been proven valid.

John P. Clark developed a solution to the equations stated earlier [2]. This culminated in the model in Figure 8.

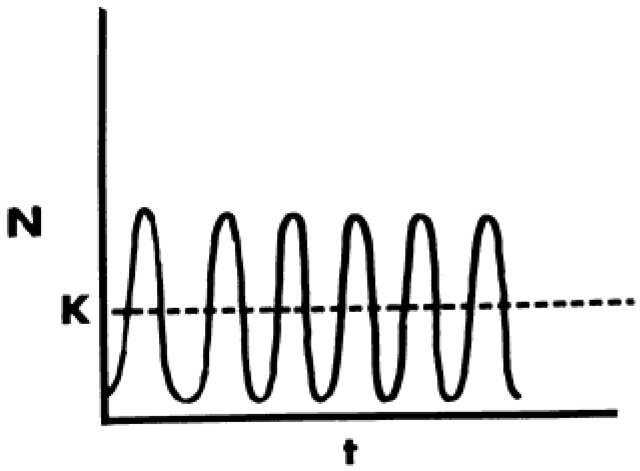


Figure 8 – Clark's model of lynx population

Clark's model is extremely similar to the model developed for this project using both ode45 and Euler's method. The amplitudes are relatively constant. The frequency is constant, and it starts near zero. Additionally, it shows the K value to be in the middle of the wave which is where it is in the models mentioned earlier. Here, N represents the number of lynx and t represents time. Clark finds not the numbers, but the "general shape of the curves" to be most important [2]. The accuracy of the shape of the model shows that the second-order equations ability to add the historical element into the model is what is most important. Therefore, according to Clark, there is "at least some doubt as to the validity of immediately beginning with first-order differential equations and assuming them to be generally applicable" [2]. The findings of this project have confirmed the beliefs expressed by Clark surrounding second-order differential equations and their ability to accurately model populations.

Conclusion

As can be seen from the Results section, the model developed by MATLAB's built in function ode45 was extremely similar to the model created through the use of Euler's method. The main difference was that ode45 was much more rigid and appeared to be missing data. Euler's method turned out to be a better modelling tool due to the smoothness of the final plot. It also picked up on trends and data that ode45 had missed. For example, the amplitude was higher in the model build with Euler's method. It was also a much cleaner sine wave. To implement Euler's method correctly, the step size needed to be 10,000. If it was a number less than this, for example 1000, the model would break. It is because the user is given more control over the variables used in Euler's method that it is able to be so much more precise, at least in this scenario.

However, these small differences don't have a large effect on the true purpose of modeling these equations: to get the general shape. This is explained by Clark and evidenced in Figure 8. Comparing all three models and the plotted real-world data, they are all extremely similar and validate the approach of using second-order equations to model population. This is because it allows for the introduction of the historical elements on the population. The derivative found in post-processing and the minima and maxima found thereafter, also proved this fact. The population is accelerating and decelerating in a way that is similar to the shape of the actual population booms.

Therefore, it can be accurately stated that the numerical techniques used to produce this project were used appropriately and successfully modelled the desired environment. The application of ode45 and Euler's method were done by first splitting the second-order equation into a system of first-order differential equations. These were then utilized to produce an "A" and "b" matrix which allowed for the solving of the system through the creation of a MATLAB anonymous function. This function was then used to model the system using ode45 and Euler's method. These models were then compared to the real-world data and deemed valid. Post-processing begun through the use of MATLAB's diff command which allowed for the plotting of the derivative through use of element-by-element division. Through this understanding, the minima and maxima were able to be analyzed. These numerical analysis methods allowed for the creation of multiple valid models which were able to confirm the results of John P. Clark's paper regarding the use of second-order differential equations in population modelling. Each model and technique was able to generate a shape that further proved his thesis. The use of second-order differential equations is vital for introducing historical effects into a system being modeled [2].

The concepts of numerical analysis through the use of ode45 and Euler's method were the most important notions utilized throughout the project. The concept of approximating the value of solutions for a differential equation was integral to understanding how to use Euler's method. Comprehending that Euler's method is a single step integration method that computes successive solution points given initially known ones was extremely important. Also, knowing how changing the step size alters the outputted model was necessary. Additionally, understanding that ode45 works for non-stiff ordinary differential equations and solves them by integrating over the given time period, given the initial conditions, was critical. Finally, realizing that the diff command can be used to differentiate two separate variables, divide them, and plot them to get the derivative was crucial.

Through the application of multiple techniques and functions learned through the semester, the model generated for this project was able to successfully and accurately produce verifiably correct results. An understanding of numerical analysis techniques, calculus, MATLAB, and the principles surrounding computer simulation was utilized and satisfactory models were produced.

References

[1] S. Ahmad, I. Stamova, and B. Lisena, "Competitive Lotka-Volterra systems with periodic coefficients," in *Lotka-Volterra and related systems recent developments in population dynamics*, Berlin: De Gruyter, 2013, pp. 63–121 [Online]. Available: https://ebookcentral-proquest-com.libdatabase.newpaltz.edu/lib/newpaltz-ebooks/reader.action?docID=1121628&ppg=71. [Accessed: 09-Mar-2022].

[2] J. P. Clark, "The second derivative and population modeling," *jstor.org*, 01-Jul-1971. [Online]. Available: https://www.jstor.org/stable/pdf/1934148.pdf. [Accessed: 09-Mar-2022].

[3] G. Innis, "The second derivative and population modeling: Another View," *jstor.org*, 01-Jul-1972. [Online]. Available: https://www.jstor.org/stable/1934789. [Accessed: 09-Mar-2022].

[4] M. L. Rosenzweig, "Why the prey curve has a hump," *jstor.org*, 01-Jan-1969. [Online]. Available: https://www.jstor.org/stable/2459470. [Accessed: 09-Mar-2022].

[5] C. Elton and M. Nicholson, "The ten-year cycle in numbers of the Lynx in Canada," *jstor.org*, 01-Nov-1942. [Online]. Available: https://www.jstor.org/stable/1358. [Accessed: 09-Mar-2022].

Appendix (CODE)

```
%Patrick Pinello / Final Project
%Predicting population booms in Lynx using
%ode45 and Eulers and diff

   %%%%% ODE45 %%%%%
   k = 37000; %mid point of population data
   a = .42; % This value determines the amount of peaks in a given time period
   b = 0; %b is stated to be zero in the paper
   tspan = [1823 1886]; %Time range (years)
   y0 = [872 1000]; %Initial conditions N2

   A = [0 1; -a -b]; %A matrix. In first row N2 is present, 2nd row represents the coef after solving for N"
   b = [0; a*k]; %column of constants
   N2 = @(t,x) A*x + 1/5*b; %anonymous function finding the vector of x derivatives
   [t,y] = ode45(N2, tspan, y0); %using ode45 to solve

   %%% Eulers %%%
   tm = tspan(2)-tspan(1); %complete time
   h = tm/10000; %step size (using 10000 (anything smaller failed)
   nSteps = tm/h; %finding number of steps to use
   [xE,yE] = odeEuler(N2,tspan(1),h,nSteps,y0); %solving using odeEuler

   %%% ORIGINAL DATA %%%
   %Lynx Population data
   lynx = [4059 2385 2208 1563 872 1510 1572 4417 10271 14135 6676 3341 2436 2420 3034 7141 14168 24788 23572 11670
3033 1867 2226 3874 7063 9082 10610 14408 3823 1007 649 658 852 3210 8668 9334 17144 7718 4372 1846 992 827 1385
3377 9743 21096 23588 15363 4780 2252 701 712 1834 3578 7235 14523 12126 6429 7072 3153 1318 1560 3587 10331];
   %%% Plotting original data %%%
   plot(tspan(1)-2:tspan(2)-2,lynx); hold on;

   %%% Plotting ode45 MODEL %%%
   plot(t,y); hold on
   % Deleting unnecessary line
   children = get(gca, 'children');
   delete(children(1));

   %%% Plotting Eulers MODEL %%%
   plot(xE,yE(:,1))
```

```
    % ALL PLOT LABELS %
    title('Population boom cycles in Lynx')
    xlabel('Years')
    ylabel('Number of Lynx')
    legend({'Real Data','Model (ode45)','Model (Eulers)'},'Location','northwest')

    figure()

    %%% Post-Processing (Derivative) %%%
    dt = diff(t); %diff of t
    dy = diff(y); %Diff of x
    dy_dt = dy./dt; %Derivative with element by elemnt
    plot(t(1:end-1),dy_dt);
    %Deleting unnecessary line
    children = get(gca, 'children');
    delete(children(1));

    %Acceleration
    title('Acceleration of Population of Lynx')
    xlabel('Years')
    ylabel('Population of Lynx / Time')

    %Directly comparing Euler's and ode45
    figure()
    plot(t,y,'r'); hold on
    % Deleting unnecessary line
    children = get(gca, 'children');
    delete(children(1));

    %%% Plotting Eulers MODEL %%%
    plot(xE,yE(:,1),'b')

    % ALL PLOT LABELS %
    title('Population boom cycles in Lynx (ode45 vs Eulers)')
    xlabel('Years')
    ylabel('Number of Lynx')
    legend({'Model (ode45)','Model (Eulers)'},'Location','northwest')


%%% odeEuler Function %%%
```

```
function [x,y] = odeEuler(f,x0,h,nSteps,y0)
% Eulers Method Initialize the output solution arrays.

x(1) = x0; %First row of x is x0
y(:,1) = y0; %first col of y is y0

% Compute x an y using Euler's method.
x = x0:h:(x0+nSteps*h); % x is the range from x0 to x0 plus the number
                        % of steps times the step count and the step
                        % interval

for i = 1:length(x)-1 %eulers in for loop % 1 to the length of x minus 1
  s = f(x(i),y(:,i)); %the function at a specific point in loop
  y(:,i+1)=y(:,i)+s*h; %function for eulers
end

y = y'; % output xm as a series of column vectors.

end
```