Lab Report 4

Team Members: Joie Green, Melissa Prado, Niva Razin, Alexander Ross

EA4 Lab Certification

We certify that the members of the team named below worked on this lab using only our own ideas, possibly with help from the TAs and our professors, specifically in the writing of the report, the programming and the analysis required for each task. We did not collaborate with any members of any other team nor did we discuss this lab or ask assistance from anyone outside of our team, the TAs and our professors. If we accessed any websites in working on these labs, these websites are explicitly listed in our report (with the exception of websites used only to get information on specific Matlab programs). We understand that there will be severe consequences if this certification is violated and there is unauthorized collaboration.

1.	Name Niva Razin	Signature [illegible]	Date 3/08/2020
2.	Name Alexander Ross	Signature Alexander Ross	Date 03/08/2020
3.	Name Melissa Prado	Signature [illegible]	Date 03/08/2020
4.	Name Joie Green	Signature [illegible]	Date 3/08/2020

PART 1

TASK 1

We use Matlab to solve and graphically represent the solutions of the initial value problem (IVP) (1):

$$\vec{X}' = B\vec{X},\ B = \begin{bmatrix} .5 & .5 & -2 \\ 2.5 & -1.5 & -2 \\ 3.5 & .5 & -5 \end{bmatrix},\ \vec{X}(0) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

and the IVP (2):

$$\vec{X}' = A\vec{X},\ A = \begin{bmatrix} 196.5 & 1.5 & -199 \\ 200.5 & -2.5 & -199 \\ 396.5 & 1.5 & -399 \end{bmatrix},\ \vec{X}(0) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

We use the *eig* function to determine the three eigenvalues of (1):

$$\lambda_1 = -200,\ \lambda_2 = -1,\ \lambda_3 = -4$$

and the corresponding eigenvectors of (1):

$$\vec{v}_1 = \begin{bmatrix} 0.4082 \\ 0.4082 \\ 0.8165 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix}, \vec{v}_3 = \begin{bmatrix} 0.5774 \\ -0.5774 \\ 0.5774 \end{bmatrix}.$$

We scale each vector by its respective smallest common denominator term:

$$\vec{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \vec{v}_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

We plug these eigenvalues and scaled eigenvectors into the general solution of $\vec{X}' = A\vec{X}$:

$$\vec{x}(t) = \alpha e^{\lambda_1 t}\vec{v}_1 + \beta e^{\lambda_2 t}\vec{v}_2 + ... + \gamma e^{\lambda_n t}\vec{v}_n,$$

with $n = 3$ for the three eigenvalues associated with $\overline{A}$:

$$\vec{X} = \alpha e^{-200t} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} + \beta e^{-t} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \gamma e^{-4t} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Then, using the Matlab backslash operator and the initial conditions of problem (1), we solve for $\alpha, \beta,$ and γ to get the exact linear combination:

$$\vec{X} = 2e^{-200t}\begin{bmatrix}1\\1\\2\end{bmatrix} - \frac{1}{2}e^{-t}\begin{bmatrix}1\\1\\1\end{bmatrix} - \frac{1}{2}e^{-4t}\begin{bmatrix}1\\-1\\1\end{bmatrix}$$

Finally, we find the solution vector $\overline{X}$ of (1) when $t = 2$ by plugging in this value:

$$\vec{X}(t=2) = \begin{bmatrix}-0.067835372932266\\-0.067499910304367\\-0.067835372932266\end{bmatrix}.$$

We repeat this process for (2), simply changing the numerical values in our code. Thus, we get the eigenvalues:

$$\lambda_1 = -3,\ \lambda_2 = -1,\ \lambda_3 = -2$$

and the same corresponding eigenvectors as we did for (1). Again scaling these eigenvectors and plugging them and the eigenvalues into the general solution, we obtain:

$$\vec{X} = \alpha e^{-3t}\begin{bmatrix}1\\1\\2\end{bmatrix} + \gamma e^{-t}\begin{bmatrix}1\\1\\1\end{bmatrix} + \beta e^{-2t}\begin{bmatrix}1\\-1\\1\end{bmatrix}$$

By the same method as before, we find the exact linear combination which gives the solution for the initial conditions of (2):

$$\vec{X} = 2e^{-3t}\begin{bmatrix}1\\1\\2\end{bmatrix} - \frac{1}{2}e^{-t}\begin{bmatrix}1\\1\\1\end{bmatrix} - \frac{1}{2}e^{-2t}\begin{bmatrix}1\\-1\\1\end{bmatrix}.$$

Again plugging in for $t = 2$, we get:

$$\vec{X}(t=2) = \begin{bmatrix}-0.071867956709340\\-0.053552317820606\\-0.066910452356008\end{bmatrix}.$$

We now plot each component of the solutions of (1) and (2) on the interval $t = 0$ to $t = 2$. We define each component as $x(t)$, $y(t)$ and $z(t)$ with the notation:

$$\vec{X}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}.$$

We additionally plot the corresponding long time behavior of each component, given by:

$$\vec{XL}(t) = \begin{bmatrix} xl(t) \\ yl(t) \\ zl(t) \end{bmatrix}.$$

The long time behavior term is defined as the term of each component with the smallest corresponding $|\lambda|$ value (the eigenvalue closest to zero). This term gives the most slowly decaying part of the solution. The plots of the components of (1) are shown in **Figures 1a-c**, and the plots of the components of (2) are shown in **Figures 2a-c**.

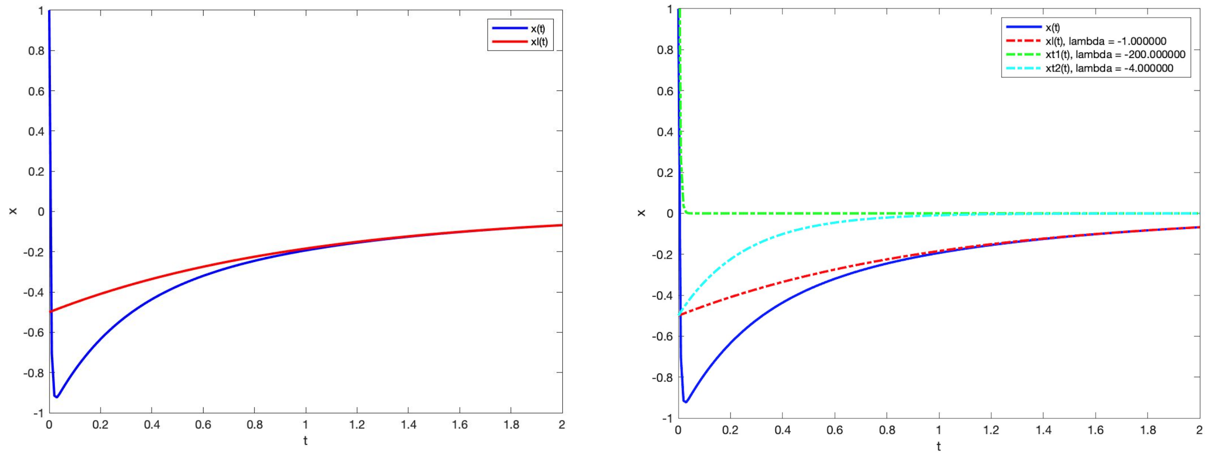


Figure 1a. Left: *overall behavior $x(t)$ in blue and long time behavior $xl(t)$ in red of IVP (1) from time $t = 0$ to $t = 2$ for the given initial conditions.* ***Right:*** *the two transient terms of the $x(t)$ component, $xt1(t)$ and $xt2(t)$, are plotted in green and cyan along with the long time and overall behavior. Note how quickly the transient terms decay to zero relative to the long time behavior term. Also, note that adding the values of $xt1(t)$, $xt2(t)$, and $xl(t)$ at each t gives the value of $x(t)$ at that point (i.e., it's a linear combination).*

We classify IVP (1) as a stiff problem based on the disparate magnitudes of its eigenvalues, 200 compared to 1 and 4. We thus see the terms of each component of the solution decay at very different rates. Conversely, the terms of each component of the solution of IVP (2), which is nonstiff based on its eigenvalues of similar magnitudes, decay at similar rates.

The plots of IVP (1) in **Figures 1a-c** manifest stiff behavior in the initial, sharp decay of the overall component (caused by the transient terms). The plots of IVP (2) in **Figures 2a-c** manifest nonstiff behavior by the more gradual decay of overall component (the decay of the transient and long term behavior terms are similar in magnitude). For all of these plots, the overall component converges on the long term behavior, as the transient terms die out quickly.

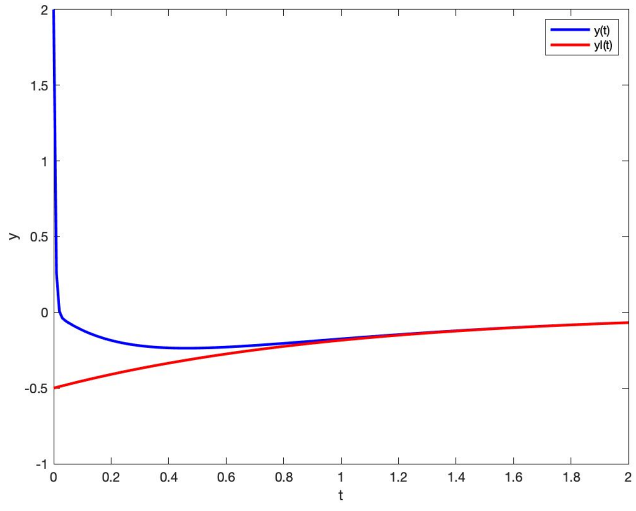


Figure 1b. *Overall behavior $y(t)$ and long-time behavior $yl(t)$ of IVP (1) from time $t = 0$ to $t = 2$ for the given initial conditions.*

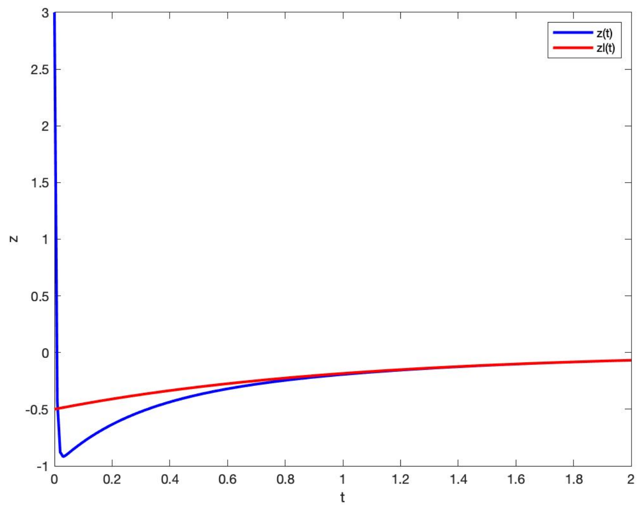


Figure 1c. *Overall behavior $z(t)$ and long-time behavior $zl(t)$ of IVP (1) from time $t = 0$ to $t = 2$ for the given initial conditions.*

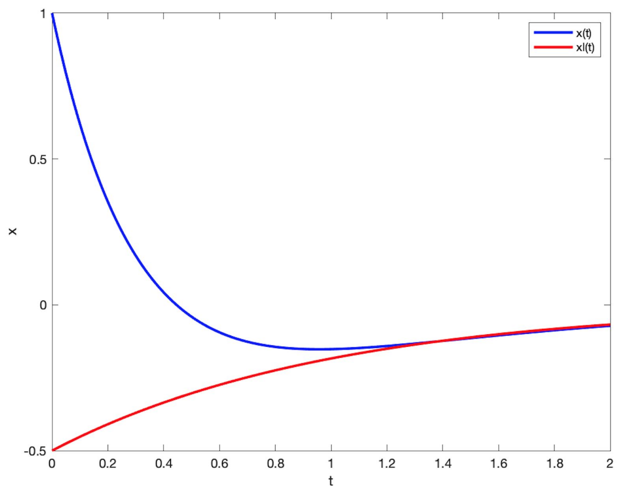


Figure 2a. *Overall behavior $x(t)$ and long-time behavior $xl(t)$ of IVP (2) from time $t = 0$ to $t = 2$ for the given initial conditions.*

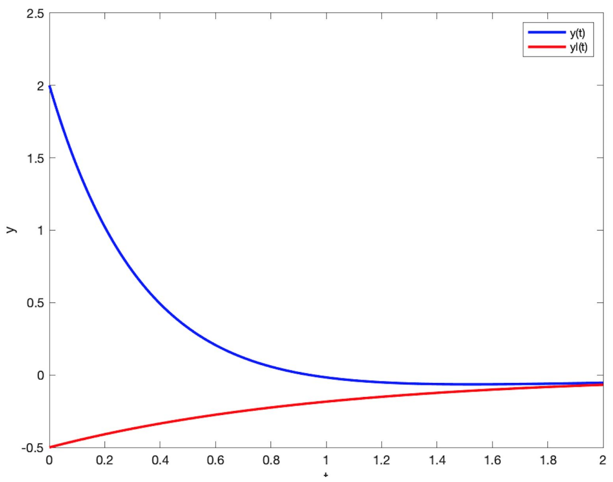


Figure 2b. *Overall behavior $y(t)$ and long-time behavior $yl(t)$ of IVP (2) from time $t = 0$ to $t = 2$ for the given initial conditions.*

The term of each component with the associated eigenvalue closest to zero ($\lambda = -1$ for both IVP (1) and (2)) is called the long time behavior or steady state regime, as it takes the longest to decay. In **Figures 1a-c** and **2a-c** this long term behavior term is plotted in red. The overall behavior of the component (the sum of the steady state and transient responses) is plotted in blue. In the right hand **Figure 1a**, the transient terms of the $x(t)$ component, denoted $xt1(t)$ and $xt2(t)$, are also plotted in cyan and green. Their transient behavior is evident by their relatively rapid decay, caused by their eigenvalues of large magnitude.

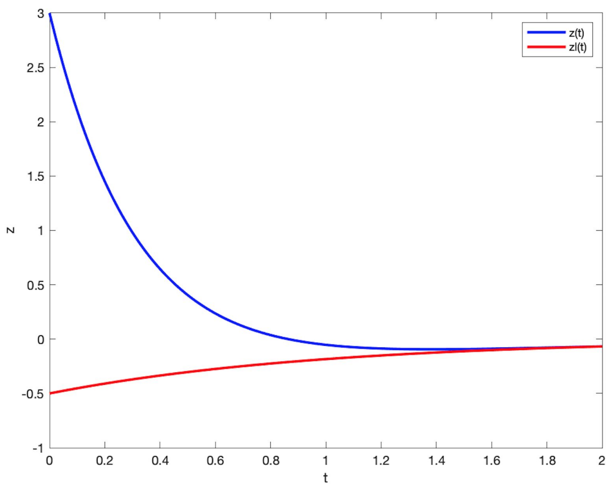
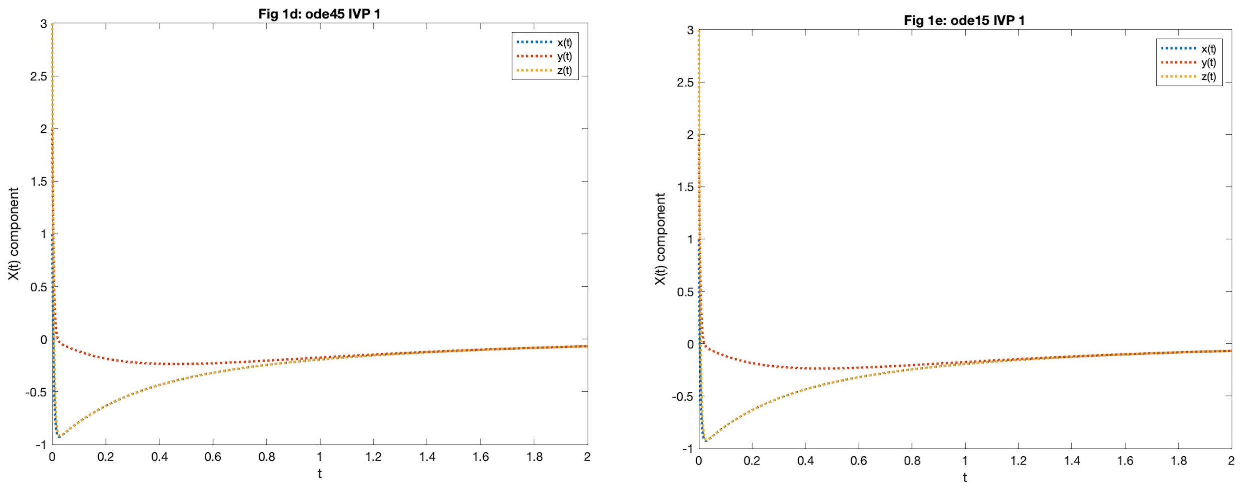
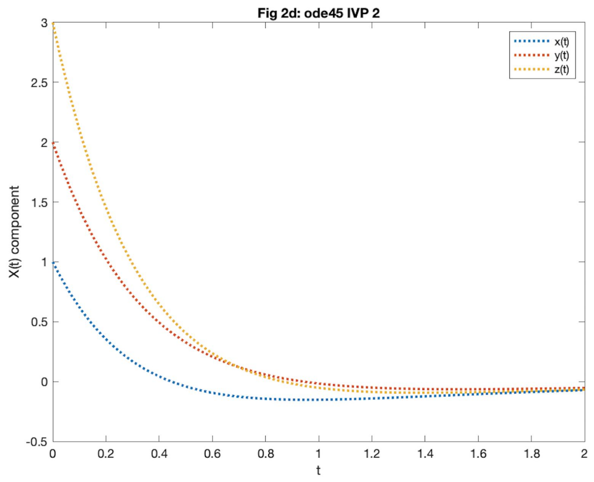


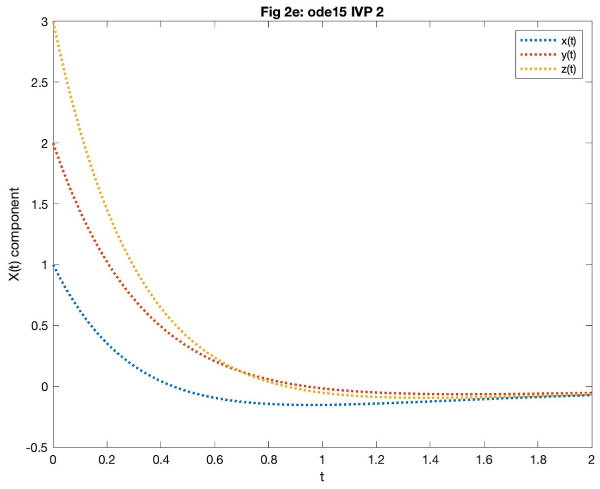
Figure 2c. *Overall behavior $z(t)$ and long-time behavior $zl(t)$ of IVP (2) from time $t = 0$ to $t = 2$ for the given initial conditions.*

TASK 2



Figures 1e-d. *The components of the solution of IVP (1), found using ode45 (left) and ode15s (right).*





Figures 2e-d. *The components of the solution of IVP (2), found using ode45 (left) and ode15s (right).*

TASK 3

We discussed in Task 1 how the difference in magnitude of eigenvalues qualifies IVP (1) to be a stiff problem and IVP (2) to be nonstiff. Based on the different time scales associated with stiff problems, we expect IVP (1) to be more difficult to compute using conventional explicit schemes.

Table 1. *Values calculated in Part 1. N15 and N45 indicate the nsteps, a measure of average timestep, for ode15 and ode45. C15 and C45 indicate count, the number of times the helper function prime.m was called. RN is the ratio N15s/N45, and RC is the ratio C15s/C45. The values of e15 and e45 represent the errors of the solutions of respective ode functions. They were calculated as the greatest difference between the solution from the ode function and the solution of Task 1 at t=2.*

Matrix	**N15s**	**C15s**	**N45**	**C45**	**RN**	**RC**	**e15**	**e45**
A IVP (1)	135	277	545	883	0.24770	0.31370	$3.83112038\text{x}10^{-7}$	$3.703204\text{x}10^{-7}$
B IVP (2)	57	119	57	85	1	1.40000	$1.04895152\text{x}10^{-6}$	$1.019738\text{x}10^{-6}$

Since ode15s is an implicit method, it is unconditionally stable. Thus, as its name indicates, it is meant to solve stiff problems, which otherwise require very small timesteps of explicit, nonstiff solvers like ode45. In **Table 1**, RN, the ratio of average timestep found in ode15 to ode45, is about 25% for (1). This indicates that the ode15s function uses much larger smaller timesteps than ode45 for this stiff problem, and thus has a greater timstep efficiency. This makes sense given ode15s uses an implicit scheme. ode45 takes more steps to reach nearly the same accuracy as ode15s achieves, as indicated by their error values. In this case ode15s is more efficient for than ode45 based on metric of computational work, as well ($RC < 1$). While the functions give nearly identical solutions (**Figure 1d** vs. **1e**), we see that ode15s is the more efficient of the two functions for this stiff problem.

For the nonstiff IVP (2) problem, ode15s has the same timestep efficiency as ode45 ($RN = 1$) but requires more computational work, indicated by the large RC ratio ($RC > 1$). As indicated by the error values, this extra work does not pay off, as the ode15s is actually slightly less accurate than the ode45 solution. Graphically, however, the solutions of the two functions are nearly identical (**Figure 2d** vs. **2e**).

We can thus conclude that the implicit solver ode15s ought to be used only for stiff problems and the explicit solver ode45 for nonstiff problems if one wants to maximize the timestep size efficiency (*RN*) and computational work efficiency (*RC*).

PART 2

TASK 1

In this part of the lab, we are asked to verify the system

$$x' = y = F(x,y)\,, \quad y' = -x - \beta x^3 = G(x,y) \qquad (1)$$

when $\beta \geq 0$.

We simplify $G(x,y)$ such that

$$G(x,y) = -x(1+\beta x^2)$$

We can see from this simple factoring process that the critical points of the system are $(x,y) = (0,0)$. Thus we can determine the Jacobian matrix of the system:

$$\tfrac{\delta F}{\delta x} = 0 \qquad \tfrac{\delta G}{\delta x} = (-1 - 3\beta x^3)$$

$$\tfrac{\delta F}{\delta y} = 1 \qquad \tfrac{\delta G}{\delta y} = 0$$

Then, to determine the trajectory, we take the determinant of the Jacobian matrix associated with the above system. Here we see that:

$$det\,(J_{(0,0)} - \lambda I) = (1+\lambda^2) = 0$$

$$\lambda = \pm i$$

Here, we see that the eigenvalues of the system are purely imaginary. Based on this result, we can classify the critical point as a linear center at the origin.

TASK 2

Here, we are asked to analyze the following system:

$$mx'' + cf(x)x' + kx = 0$$

We are asked to take $c > 0$ and $m = k = 1$, resulting in

$$x'' + cf(x)x' + x = 0$$

We start by writing the following system as a first order system:

$$x' = y = F(x,y)\ ,\ y' = \ x'' \ = (-\,cf(x)x' + x) = (-\,cf(x)y + x) \ = G(x,y)$$

By analyzing the above expressing, we can see that the only critical point that satisfies both equations is $(x,y) \ = \ (0,0)$. This is true for both $f(x) = 1$ and $f(x) = (-\,1)$. Now, we will analyze the first case ($f(x) = 1$) to determine the nature of the trajectories of that system. To do so, we first determine the Jacobian matrix of the system:

$$\tfrac{\delta F}{\delta x} = 0 \qquad \tfrac{\delta G}{\delta x} = 1$$

$$\tfrac{\delta F}{\delta y} = 1 \qquad \tfrac{\delta G}{\delta y} = (-\,c)$$

Then, to determine the trajectory, we take the determinant of the Jacobian matrix associated with the above system. Here we see that:

$$det\,(J_{(0,0)} - \lambda I) = (-\,c - \lambda)(-\,\lambda) - 1 = 0$$

$$\lambda^2 + c\lambda - 1 = 0$$

$$\lambda = \tfrac{-c \pm \sqrt{c^2 - 4}}{2}$$

Here, we see that there are two different possibilities for the eigenvalues of the system: a purely real eigenvalue solution when $c > 2$, and a solution of the form $a \pm bi$ when $c < 2$. For the first possibility, we note that the critical point is a stable node, as the eigenvalue solution is purely real and negative in sign. For the second possibility, we note that the critical point is a stable spiral, as the eigenvalue solution is partially real and partially imaginary, with the real component being negative in sign.

Now, we will analyze the second case ($f(x) = (-\,1)$) to determine the nature of the trajectories of that system. To do so, we first determine the Jacobian matrix of the system:

$$\tfrac{\delta F}{\delta x} = 0 \qquad \tfrac{\delta G}{\delta x} = 1$$

$$\tfrac{\delta F}{\delta y} = 1 \qquad \tfrac{\delta G}{\delta y} = (c)$$

Then, to determine the trajectory, we take the determinant of the Jacobian matrix associated with the above system. Here we see that:

$$det\,(J_{(0,0)} - \lambda I) = (c - \lambda)(-\,\lambda) - 1 = 0$$

$$\lambda^2 - c\lambda - 1 = 0$$

$$\lambda = \frac{c \pm \sqrt{(c)^2 - 4}}{2}$$

Here, we see that there are two different possibilities for the eigenvalues of the system: a purely real eigenvalue solution when $c > 2$, and a solution of the form $a \pm bi$ when $c < 2$. For the first possibility, we note that the critical point is an unstable node, as the eigenvalue solution is purely real and positive in sign. For the second possibility, we note that the critical point is an unstable spiral, as the eigenvalue solution is partially real and partially imaginary, with the real component being positive in sign.

TASK 3

In this task, we study limit cycles numerically using programs *adaptlc.m* and *prime.m*. We do this by making simple modifications to *adapt.m* from Lab 3 and altering *prime.m* to solve

$$x' = y,\ y' = \ -x - cf(x)y$$

We ran the code for cases (i) x(0)=0.1, y(0)=0; (ii) x(0)=0, y(0)=3.5; (iii) x(0)=3.5, y(0)=3.5. The x(t) from all three cases are then plotted together on the same figure against time.

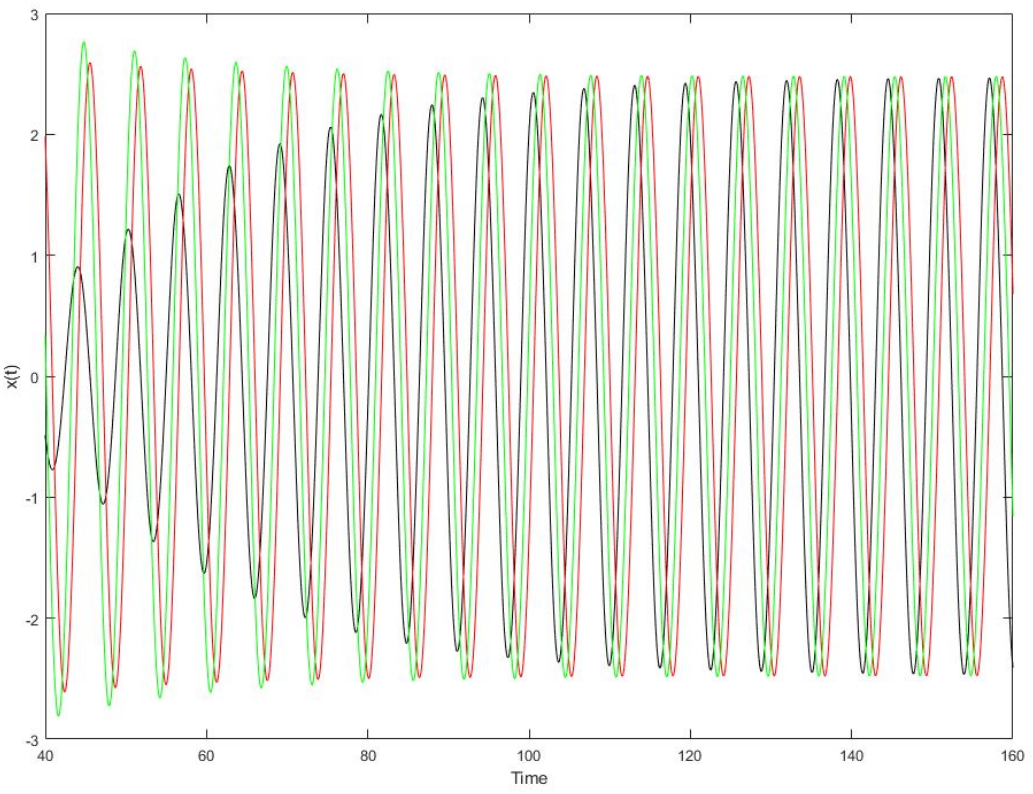


Figure 3a. *Plot of x(t) against time for cases (i) x(0)=0.1, y(0)=0; (ii) x(0)=0, y(0)=3.5; (iii) x(0)=3.5, y(0)=3.5. Case (i) is black, (ii) is red, and (iii) is green.*

As can be seen in Figure 3a, case (i) grows in time while (ii) and (iii) both decay early on. This is because in case (i) the trajectory approaching the limit cycle is a stable spiral, while for (ii) and (iii) the trajectories approaching the limit cycles are unstable spirals.

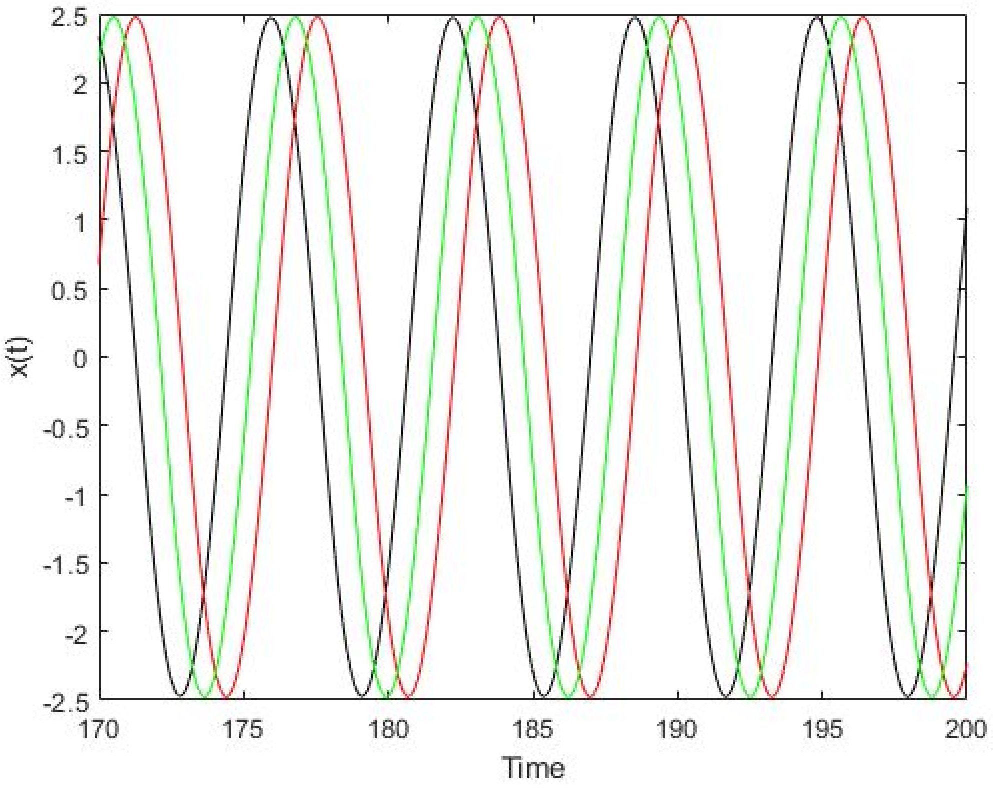


Figure 3b. *Plot of x(t) against time for the three cases during steady oscillation. Case (i) is black, (ii) is red, and (iii) is green.*

As can be seen in Figure 3b, the three curves approach the same oscillation but with different phases. The difference in phase shift is telling us that the only difference in the curves is that they are at different times on the limit cycle. That is, they approach the same limit cycle but they are at different points at different times because of their initial conditions.

In order to better visualize the trajectories approaching the limit cycle, the phase plot for each case was created. For each plot, the critical point is designated by the red dot and the initial condition is designated by the green dot.

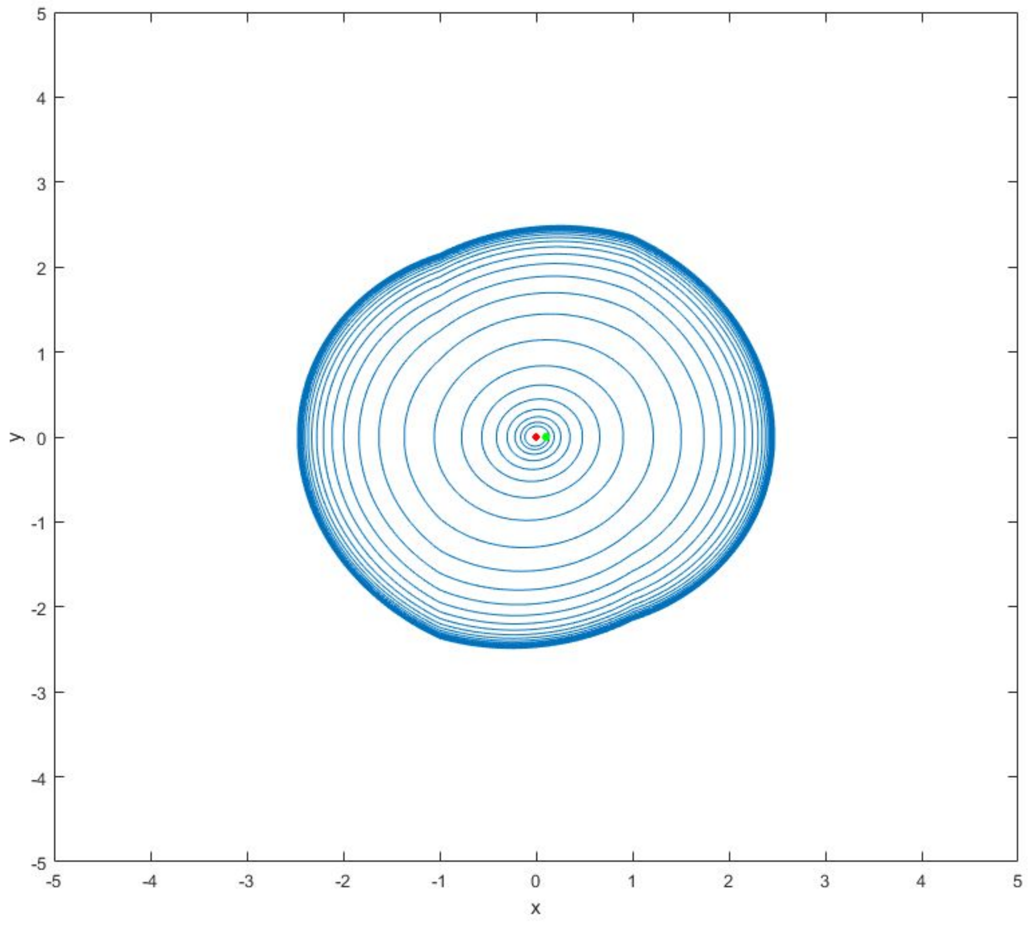


Figure 4a. *Phase plot for case (i) with red dot indicating origin and green dot indicating initial condition.*

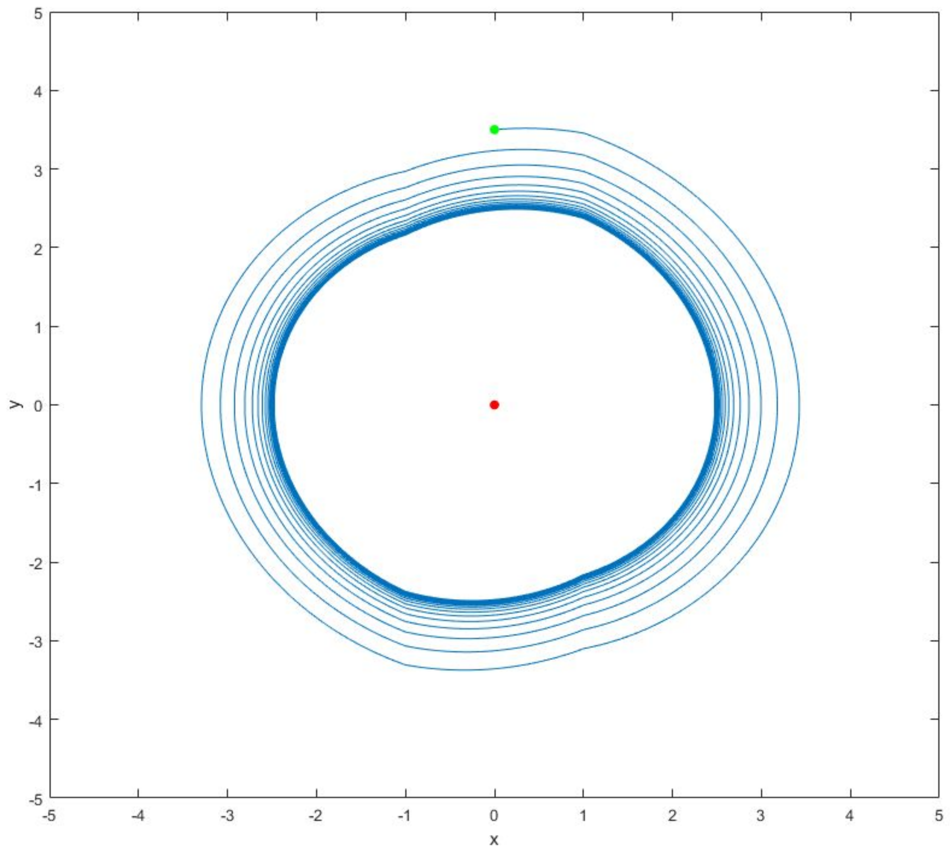


Figure 4b. *Phase plot for case (ii) with red dot indicating origin and green dot indicating initial condition.*

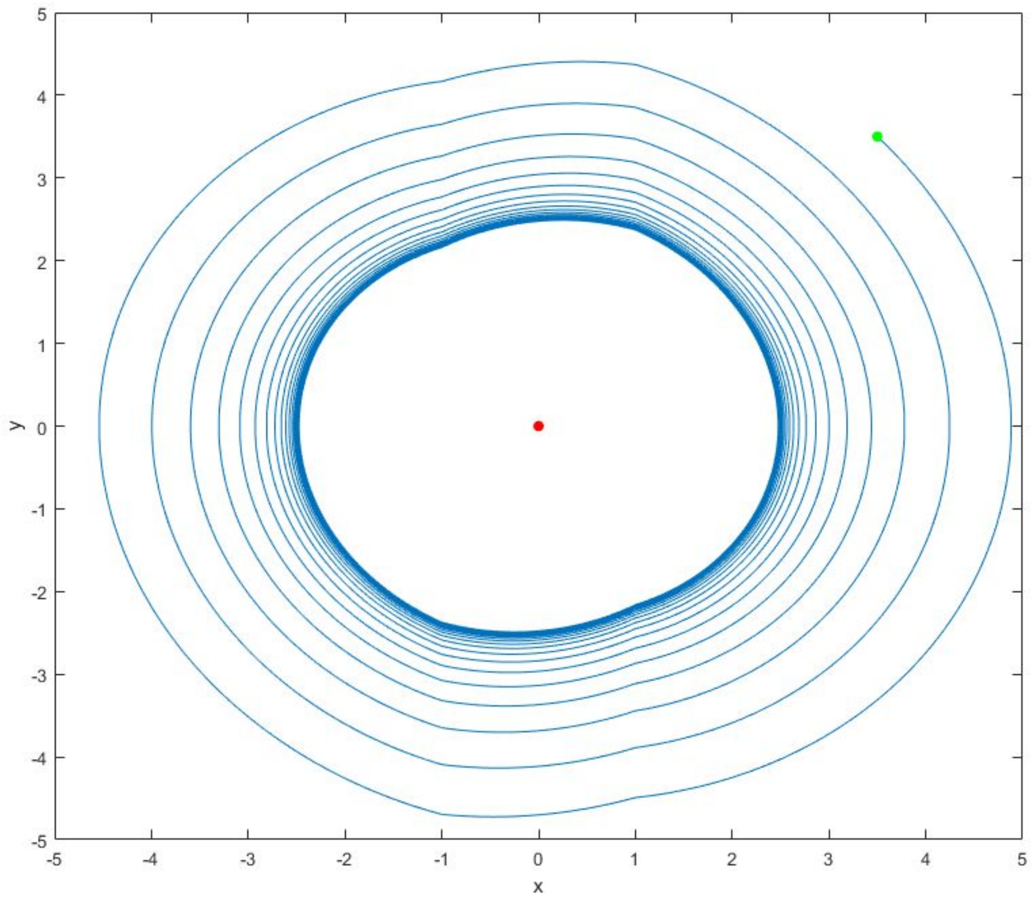


Figure 4c. *Phase plot for case (ii) with red dot indicating origin and green dot indicating initial condition.*

These plots allow us to better see that in case (i), the trajectory is a stable spiral that approaches the limit cycle from inside the closed curve. This is easy to see because the green dot, which is the initial condition, begins near the critical point and spirals away from the point toward the limit cycle. In cases (ii) and (iii), the trajectory is outside of the limit cycle and spirals toward the critical point.

For the next part of the lab, we set c = 1 and run the program *adaptlc.m* with initial conditions from case (i). We generated a phase plot of the solution, as well as a plot of X(t) and y(t) against time for the time interval 150 to 200.

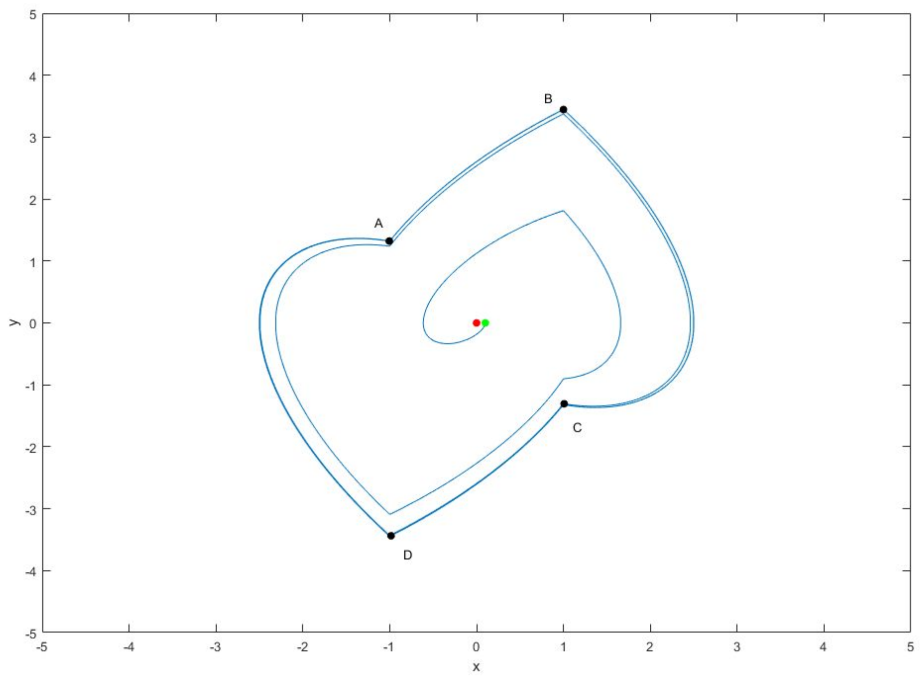


Figure 5a. *Phase plot of case (i) for c=1. Origin is designated by the red dot, the initial condition is designated by the green dot, and the three "kinks" in the limit cycle are labelled counterclockwise.*

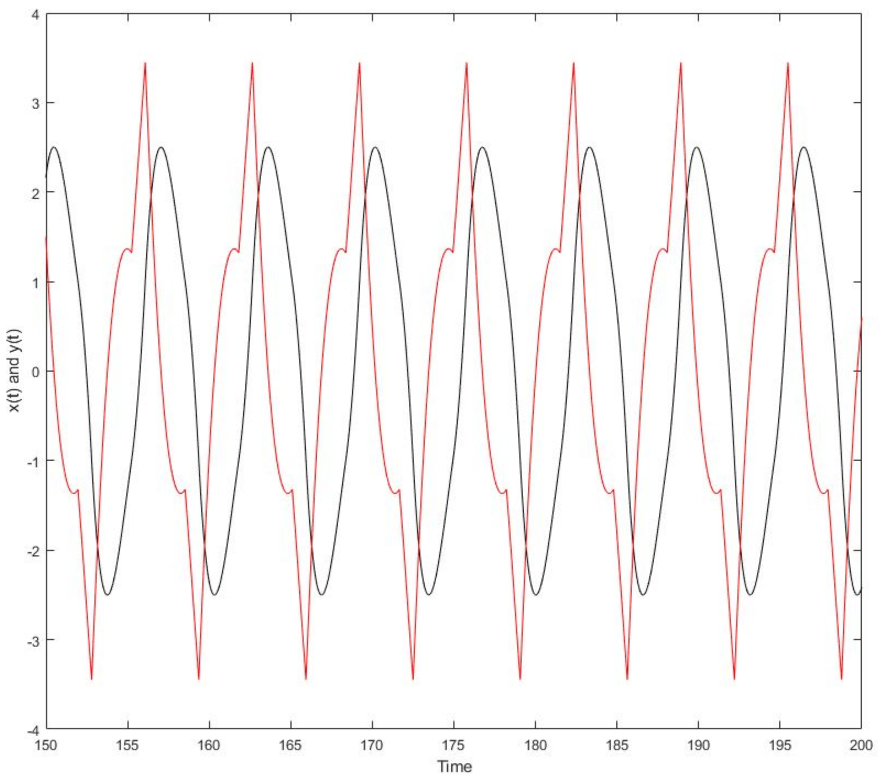


Figure 5b. *Plot of x(t) and y(t) for the time interval 150 to 200. The black curve is x(t) and the red curve is y(t)*

From Figure 5b, we can say that we have reached the limit cycle for this time interval because of the steady oscillations created by the curves with the single difference of a phase shift despite that in theory the trajectories will only asymptote to the limit cycle as time approaches infinity.

X(t) and y(t) are then plotted for a shorter interval, as shown in Figure 5c below.

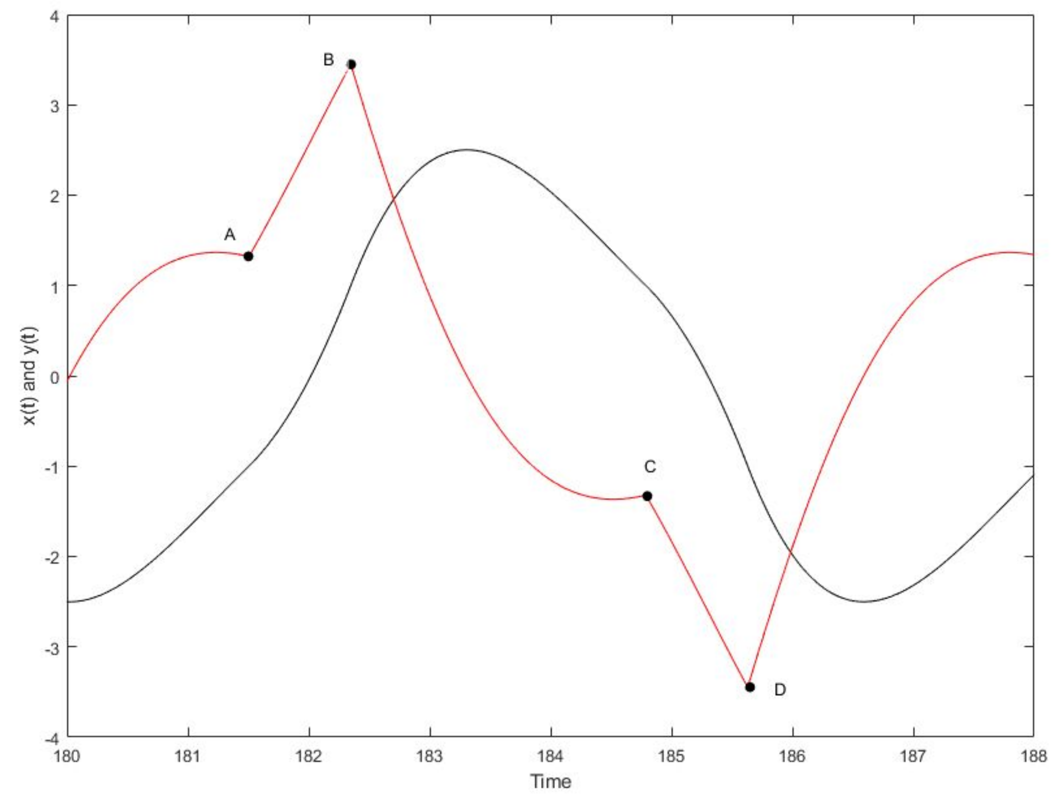


Figure 5c. *Plot of x(t) and y(t) on a shorter time interval with the kinks corresponding to Figure 5a labelled. The black curve is x(t) and the red curve is y(t).*

Each of the labelled kinks correspond to the same ones from the phase plot in Figure 5a. This is because these are the points in the solution where the function f(y) changes values.

When considering stiffness, it is easy to tell from Figure 5c that the solutions vary greatly in their timescales, which as explained in Part 1 of this lab is a sign of stiffness. The vertical jumps in y(t) are very steep, and also correspond to distortions that can be observed in x(t). X(t) is very similar to a trig function when observed on a large time interval as seen in Figure 5b, while on a short time interval the distortions are clear. This is because the kinks are the points where f(y) changes value, thereby affecting the curve x(t), and in between these points the solution behaves like a regular oscillator.

The timestep h multiplied by a factor of 100 as a function of time is then superimposed on top of the previous plot.

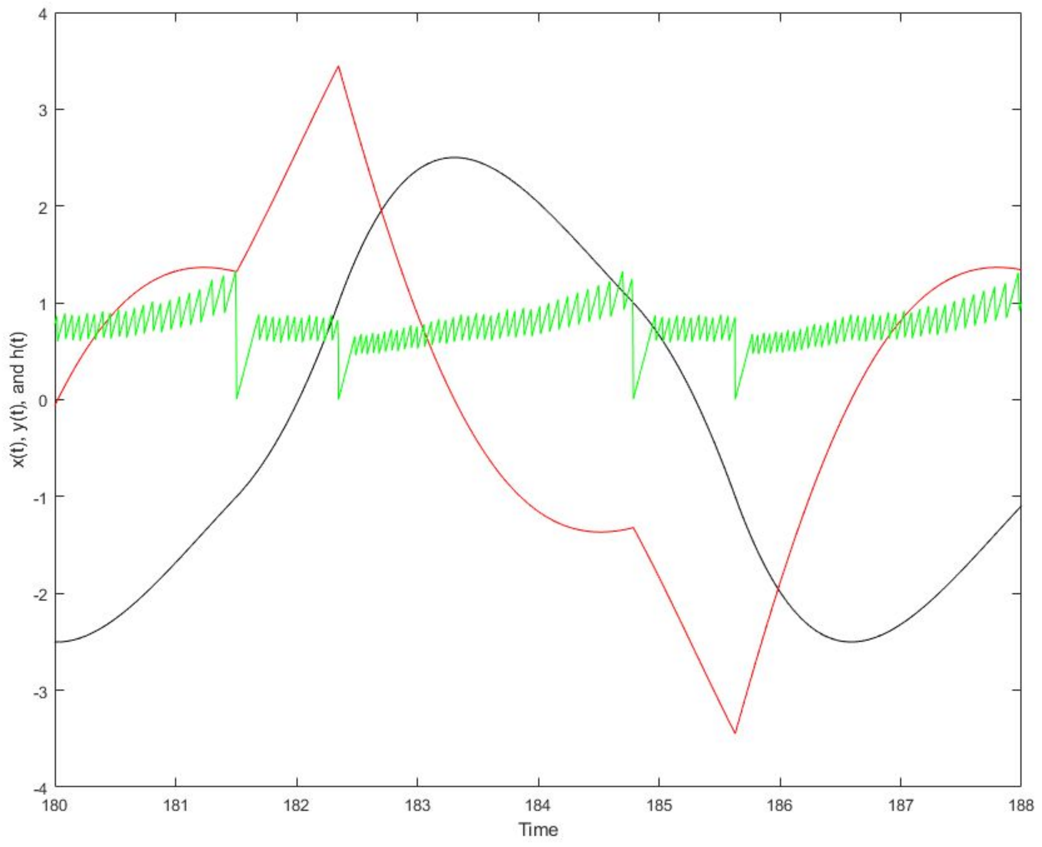


Figure 5d. *Plot of x(t) and y(t) on a shorter time interval along with timestep h multiplied by a factor of 100. The black curve is x(t), the red curve is y(t), and the green oscillation is the factored time step h.*

Figure 5d illustrates how the time step h drops significantly in magnitude at points corresponding to the kinks in y(t). This means that the program has the most difficulty at these points.

The program is then run with c=1 and with initial conditions from case (iii), and the phase plot is created.

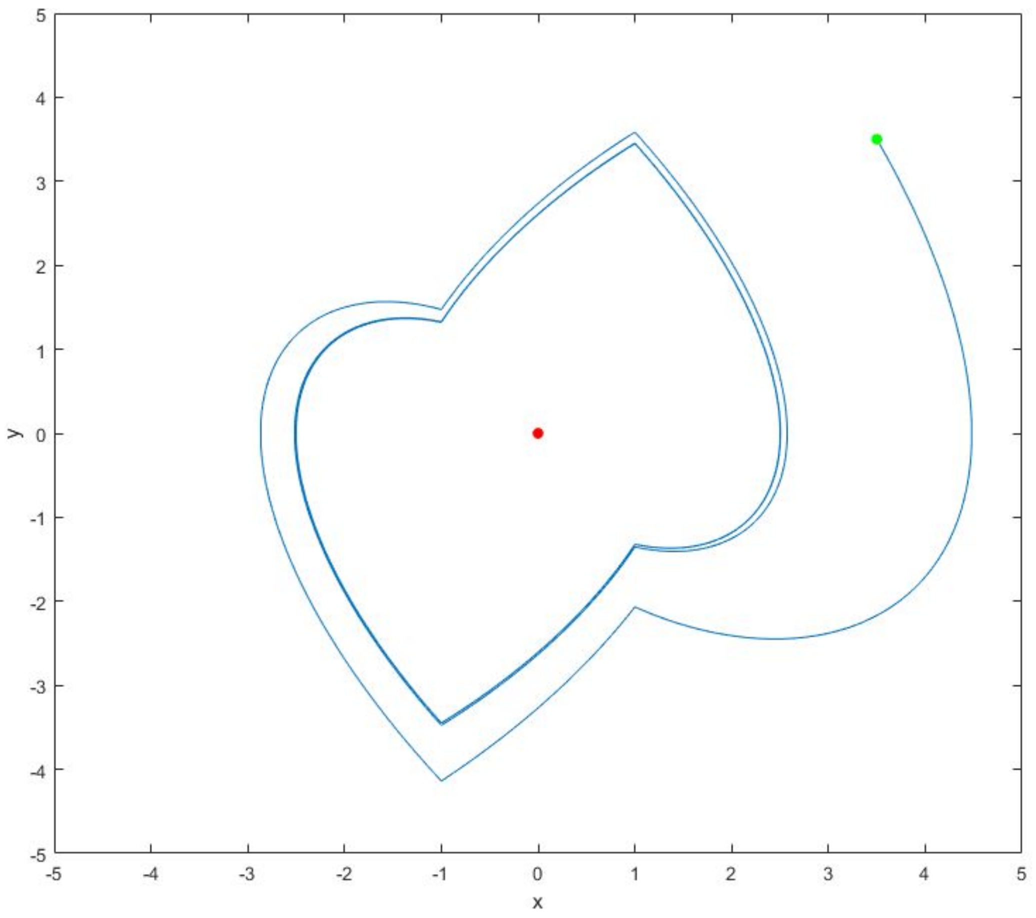


Figure 5e. *Phase plot of case (iii) with c=1. The red dot indicates the origin, and the green dot indicates the initial condition.*

As can be seen in Figure 5e, the solution asymptotes to the same limit cycle from case (i), but from the outside rather than the inside.

X(t) for both cases (i) and (ii) are then plotted against time together.

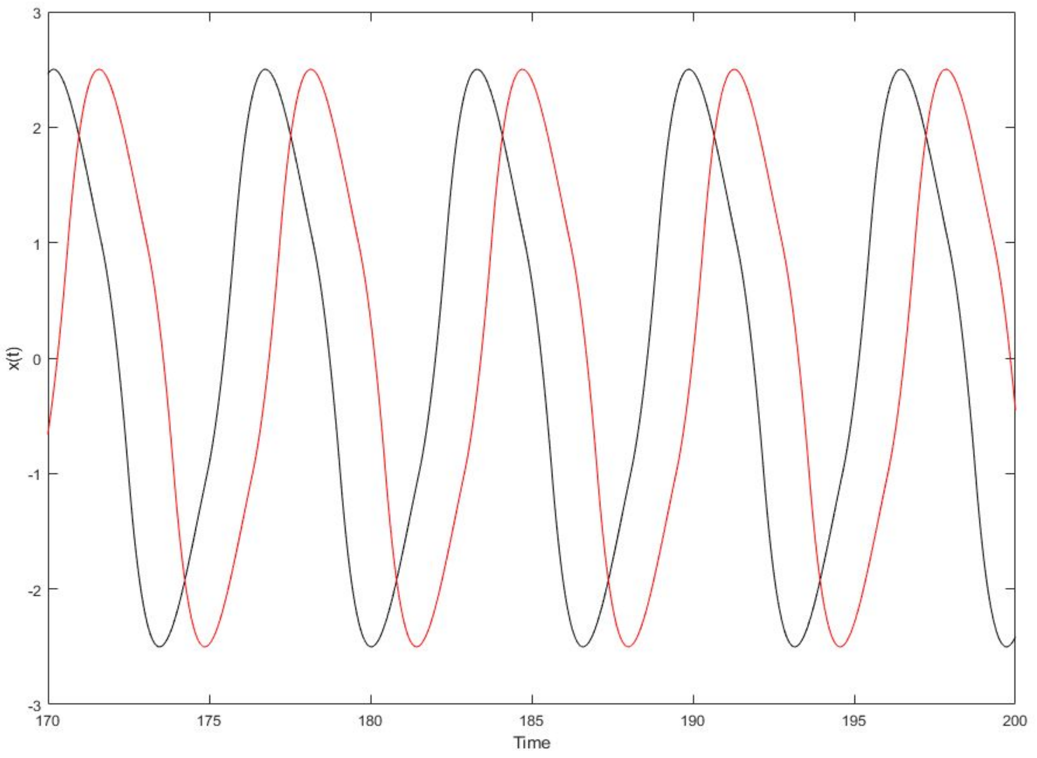


Figure 5f. *Plot of x(t) for cases (i) and (iii) where c=1 over the time interval 170 to 200 The black curve is case (i) and the red curve is case (iii).*

In Figure 5f, it can again be seen that after time the curves form the same steady oscillation with only a phase shift differentiating the two cases.

TASK 4

Here, we are asked to consider two additional nonlinear systems:

$$f(x) = -1\,,\ |x| \le 5\ ;\ f(x) = 1\,,\ |x| \ge 1.5\,;\ f(x) = 2(x-0.5)-1\,,\ 5 > |x| > 1.5 \qquad (9)$$

$$f(x) = x^2 - 1 \qquad (10)$$

We are then asked to make phase plots of these systems by modifying the code established previously in Task 3. These phase plots were generated and can be seen below:

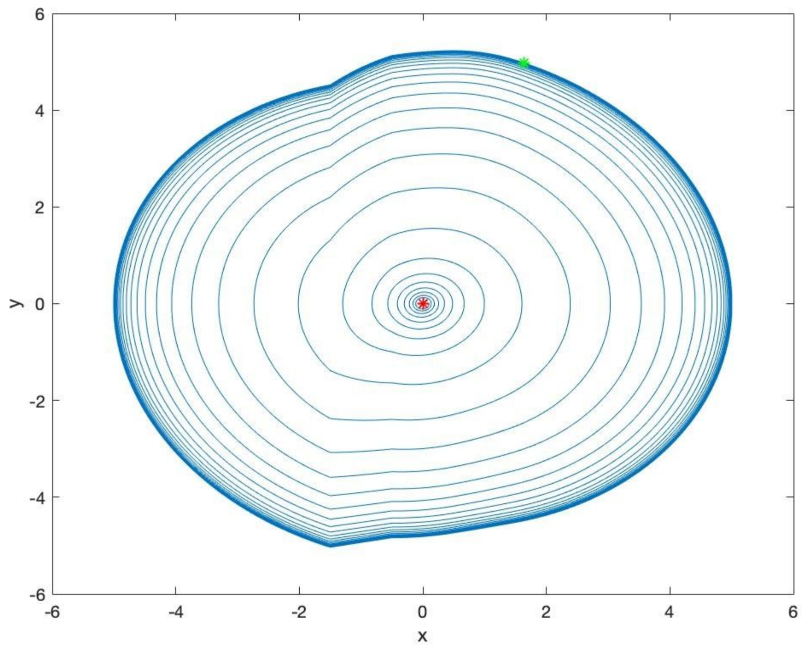


Figure 6a. *Phase plot for equation (9) with red dot indicating origin and green dot indicating initial condition.*

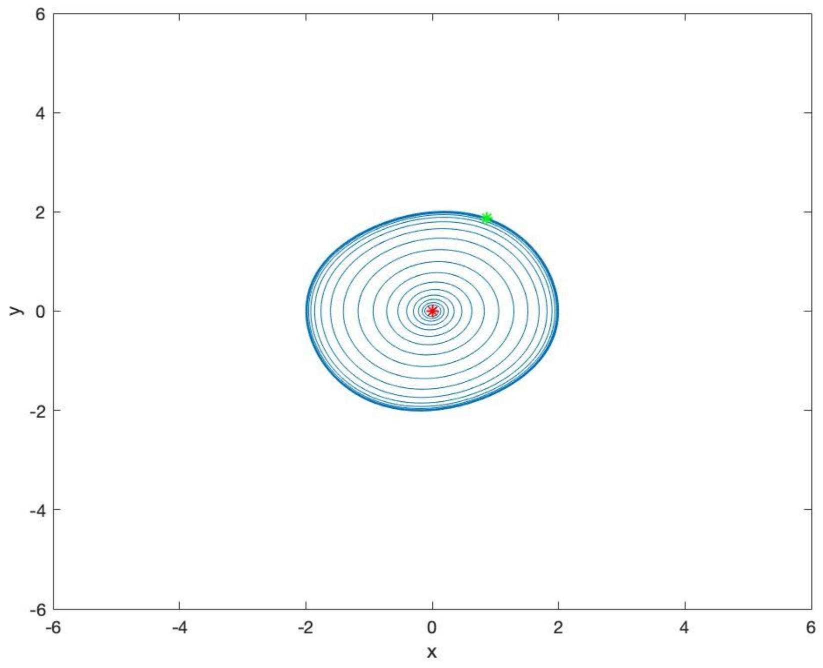


Figure 6b. *Phase plot for equation (10) with red dot indicating origin and green dot indicating initial condition.*

Here, we see that these functions are both consistent with the mechanism for limit cycles. For both equations, the trajectory is a stable spiral that approaches the limit cycle from inside the closed curve. This is easy to see because the green dot, which is the initial condition, begins far away from the critical point and spirals towards the critical point, away from the limit cycle.

<u>TASK 5</u>

In Part II of this lab, we synthesized our knowledge of the three types of oscillations that are prevalent in engineering: free oscillations, forced oscillations, and self-sustained oscillations (limit cycles). Using mathematics and our engineering intuition, we were able to classify each of these oscillations, noting the important differences between each oscillation type. In class, we learned about forced oscillations. Forced oscillations are driven by an external forcing function, and characteristic of a damped oscillator. They are manifested by a transient, depending on the initial condition, which subsequently decays at time approaches infinity.

Then, in lab 3, we extended our knowledge by considering free oscillations. Free oscillations are denoted in a phase plot by level curves around a center point, which signifies the total conservation of mechanical energy. The level curves are controlled by the initial amplitude 'A', which correspond to different periodic solutions. These oscillations don't have any forcing function, and are often characteristic of an undamped mechanical system. Finally, in this lab, we were introduced to limit cycles. Limit cycles are closed curves in which at least one trajectory spirals into it as time approaches positive or negative infinity. Limit cycles are independent of initial conditions, are only dependent on the phase angle, and are characteristic of nonlinear systems.

We subsequently analyzed several nonlinear systems both mathematically and computationally, allowing us to develop phase plots of the self-sustaining oscillations that they produced. These plots allowed us to draw conclusions about how limit cycles arise and what their properties are. We saw that limit cycles arise in an engineering when a system with no external forcing has one or more nonlinear devices or elements that is capable of storing energy when the oscillations are large and releasing stored energy when the oscillations are small. The nonlinear term is the contained in the second term of the general Liénard equation $mx'' + cf(x)x' + kx = 0$. For real systems, the $f(x)$ term is positive, which is indicative of a damping factor that causes energy dissipation after long time intervals. This is intuitive as the energy of all real systems eventually decays to zero, as energy is not completely conserved.

Mathematically, limit cycles correspond to a family of periodic solutions that differ only in the phase angle that they exist at. As a result, for many engineering purposes, this family of solutions can be thought of as a single oscillation. Additionally, limit cycle trajectories approach the limit cycle asymptotically as time goes to infinity, but never hit the limit cycle in finite time. However, for engineering purposes, when these trajectories get close enough to the limit cycle, they can represent a steady state oscillation. These are important considerations when analyzing limit cycles, as understanding the mechanism of limit cycles is fundamental to applying this knowledge to solve problems in engineering.

Limit cycles are ideal for modeling systems such as dynamic biological systems. One example of this is a beating heart. A heart can be modeled as a complex electrical device that coordinates atrial and ventricular movements by timed electrical stimuli. These stimuli are timed precisely, so that the atria and ventricles contract in synchrony with one another. In this case, the strength of the signal and the polarization of the signal can be used to describe the electrical stimuli, and together form a second order system. In this model, the heart beats periodically and nonlinearly, as the heart must variably adjust the rate at which it delivers stimuli to the atria and ventricles. This is dependent on the amount of physical stress that is placed on the body. This type of regulation is entirely self sustaining and is controlled directly by the heart, preventing external stimuli from exciting the cardiac tissue. Because the heart operates in this way, it produces self-sustaining oscillations. This is an important consideration in biomedical engineering design.

Similarly, limit cycles can be useful in mechanical systems design as well, such as in the case of a nonlinear controller. Nonlinear controllers are attractive because they are simpler to implement, respond faster, are more accurate, and reduce the energy necessary to power them. This is because they are self-regulated by self-sustaining oscillations. An example of such a controller is a thermostat-controlled heating system. In this case, the thermostat self-regulates the controlled activation of the heating system by assessing the temperature in the room in a continuous fashion. In this way, the controller is entirely self-sustaining and nonlinear, making it ideal for novel and useful purposes such as the heating of an office building. Thus, we can see that limit cycles are widely applicable to different systems and are useful in engineering design.

APPENDIX A: Part 1 Code

%%% EXACT.M %%%

```
% EA 4 LAB 4 PART 1 TASK 1

% PROBLEM 1

clear;
clc;
% X' = AX, initial conditions: X(0) = [1;2;3]
% X = (x;y;z)
A = [196.5,1.5,-199;200.5,-2.5,-199;396.5,1.5,-399];
X = [1;2;3];
[Aeig_vec,Aeig_val] = eig(A);
Aeig_val;
Aeig_vec
% scaling eigenvector values
Aeig_vec_scaled = zeros(length(Aeig_vec));
for ii=1:length(Aeig_vec)
   Aeig_vec_scaled(:,ii)=Aeig_vec(:,ii)./Aeig_vec(1,ii);
end
Aeig_vec_scaled

coeffs = (Aeig_vec_scaled\X);

% finding solution vector X at t=2
t = 2;
A_x_vec = zeros(3,1);
for j=1:3
   A_x_vec(j) = 2*exp(t*Aeig_val(1,1))*Aeig_vec_scaled(j,1) -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(j,2)
-0.5*exp(t*Aeig_val(3,3))*Aeig_vec_scaled(j,3);
end

% format long
A_x_vec % correct


% PLOT FIGURES
t_vec=0:.01:2;

% Figure 1a
x_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
   t=t_vec(i);
   x_soln(i) = 2*exp(t*Aeig_val(1,1))*Aeig_vec_scaled(1,1) -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(1,2)
-0.5*exp(t*Aeig_val(3,3))*Aeig_vec_scaled(1,3);
end
xl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
   t=t_vec(i);
   xl_soln(i) = -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(1,2);
end
xt1_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
   t=t_vec(i);
   xt1_soln(i) = 2*exp(t*Aeig_val(1,1))*Aeig_vec_scaled(1,1);
end
xt2_soln = zeros(1,length(t_vec));
```

```
for i=1:length(t_vec)
    t=t_vec(i);
    xt2_soln(i) = -0.5*exp(t*Aeig_val(3,3))*Aeig_vec_scaled(1,3);
end

eigl =round(Aeig_val(2,2));
eigt1 =(Aeig_val(1,1));
eigt2 =(Aeig_val(3,3));

plot(t_vec,x_soln, 'b','LineWidth', 2); hold on; plot(t_vec,xl_soln,'-.r', 'LineWidth', 2);plot(t_vec,xt1_soln, '-.g', 'LineWidth', 2);plot(t_vec,xt2_soln, '-.c',
'LineWidth', 2); hold off
ylim([-1 1]);
xlabel('t')
ylabel('x')
legend('x(t)', sprintf('xl(t), lambda = %f', eigl), sprintf('xt1(t), lambda = %f',eigt1), sprintf('xt2(t), lambda = %f',eigt2));
% saveas(gcf,'ea4lab4fig1a2.png')

% Figure 1b
y_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    y_soln(i) = 2*exp(t*Aeig_val(1,1))*Aeig_vec_scaled(2,1) -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(2,2)
-0.5*exp(t*Aeig_val(3,3))*Aeig_vec_scaled(2,3);
end
yl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    yl_soln(i) = -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(2,2);
end
figure();
plot(t_vec,y_soln, 'b','LineWidth', 2); hold on; plot(t_vec,yl_soln, 'r', 'LineWidth', 2); hold off
xlabel('t')
ylabel('y')
legend('y(t)', 'yl(t)');
% saveas(gcf,'ea4lab4fig1b.png')

% Figure 1c
z_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    z_soln(i) = 2*exp(t*Aeig_val(1,1))*Aeig_vec_scaled(3,1) -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(3,2)
-0.5*exp(t*Aeig_val(3,3))*Aeig_vec_scaled(3,3);
end
zl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    zl_soln(i) = -0.5*exp(t*Aeig_val(2,2))*Aeig_vec_scaled(3,2);
end
figure();
plot(t_vec,z_soln, 'b','LineWidth', 2); hold on; plot(t_vec,zl_soln, 'r', 'LineWidth', 2); hold off
xlabel('t')
ylabel('z')
legend('z(t)', 'zl(t)');
% saveas(gcf,'ea4lab4fig1c.png')
% close all;

%%
% PROBLEM 2

clear;
clc;
```

```
% X' = BX, initial conditions: X(0) = [1;2;3]
% X = (x;y;z)
B = [.5,.5,-2;2.5,-1.5,-2;3.5,.5,-5];
X = [1;2;3];
[Beig_vec,Beig_val] = eig(B);
Beig_val;
Beig_vec;
% scaling eigenvector values
Beig_vec_scaled = zeros(length(Beig_vec));
for ii=1:length(Beig_vec)
    Beig_vec_scaled(:,ii)=Beig_vec(:,ii)./Beig_vec(1,ii);
end
Beig_vec_scaled;

coeffs = (Beig_vec_scaled\X);

% finding solution vector X at t=2
t = 2;
B_x_vec = zeros(3,1);
for j=1:3
    B_x_vec(j) = 2*exp(t*Beig_val(1,1))*Beig_vec_scaled(j,1) -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(j,2)
-0.5*exp(t*Beig_val(3,3))*Beig_vec_scaled(j,3);
end

format long
B_x_vec




% PLOT FIGURES
t_vec=0:.01:2;

% Figure 2a
x_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    x_soln(i) = 2*exp(t*Beig_val(1,1))*Beig_vec_scaled(1,1) -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(1,2)
-0.5*exp(t*Beig_val(3,3))*Beig_vec_scaled(1,3);
end
xl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    xl_soln(i) = -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(1,2);
end
plot(t_vec,x_soln, 'b','LineWidth', 2); hold on; plot(t_vec,xl_soln, 'r', 'LineWidth', 2); hold off
xlabel('t')
ylabel('x')
legend('x(t)', 'xl(t)');
% saveas(gcf,'ea4lab4fig2a.png')

% Figure 2b
y_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    y_soln(i) = 2*exp(t*Beig_val(1,1))*Beig_vec_scaled(2,1) -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(2,2)
-0.5*exp(t*Beig_val(3,3))*Beig_vec_scaled(2,3);
end
yl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
```

```
    t=t_vec(i);
    yl_soln(i) = -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(2,2);
end
figure();
plot(t_vec,y_soln, 'b','LineWidth', 2); hold on; plot(t_vec,yl_soln, 'r', 'LineWidth', 2); hold off
xlabel('t')
ylabel('y')
legend('y(t)', 'yl(t)');
% saveas(gcf,'ea4lab4fig2b.png')

% Figure 2c
z_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    z_soln(i) = 2*exp(t*Beig_val(1,1))*Beig_vec_scaled(3,1) -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(3,2)
-0.5*exp(t*Beig_val(3,3))*Beig_vec_scaled(3,3);
end
zl_soln = zeros(1,length(t_vec));
for i=1:length(t_vec)
    t=t_vec(i);
    zl_soln(i) = -0.5*exp(t*Beig_val(2,2))*Beig_vec_scaled(3,2);
end
figure();
plot(t_vec,z_soln, 'b','LineWidth', 2); hold on; plot(t_vec,zl_soln, 'r', 'LineWidth', 2); hold off
xlabel('t')
ylabel('z')
legend('z(t)', 'zl(t)');
% saveas(gcf,'ea4lab4fig2c.png')
close all;
```

%%% ODE45STIFF.M %%%

```
global count;
count=0;
tbeg = 0.;
tend = 2.;
Xinit = [1;2;3];
options = odeset('RelTol', 1.75e-4); % Problem 1
% options = odeset('RelTol', 1.50e-4) % Problem 2

[t,x] = ode45(@prime,[tbeg,tend],Xinit,options);
plot(t,x);

count

% paste the following in command window after running exact.m and odestiff45.m
% abs(transpose(x(545,:))-A_x_vec) % Problem 1
% abs(transpose(x(57,:))-B_x_vec) % Problem 2
```

%%% ODE15STIFF.M %%%

```
global count;
count=0;
tbeg = 0.;
tend = 2.;
Xinit = [1;2;3];
```

```
% options = odeset('RelTol', 5e-6); % Problem 1, ode15
options = odeset('RelTol', 1.50e-5) % Problem 2, ode15

[t,x] = ode15s(@prime,[tbeg,tend],Xinit,options);
plot(t,x);

plot(t,x,':','Linewidth',2);

% title('Fig 2e: ode15 IVP 2');
xlabel('t')
ylabel('X(t) component')
legend('x(t)', 'y(t)','z(t)');
% saveas(gcf,'ea4lab4fig2e.png')

count

% paste the following in command window after running exact.m and odestiff45.m
% abs(transpose(x(545,:))-A_x_vec) % Problem 1
% abs(transpose(x(57,:))-B_x_vec) % Problem 2
```

%%% PRIME.M %%%

```
function f = prime(t,X)
global count;
count = count+1;

global A;
A = [196.5,1.5,-199;200.5,-2.5,-199;396.5,1.5,-399]; % Problem 1
% A = [.5,.5,-2;2.5,-1.5,-2;3.5,.5,-5]; % Problem 2
f = (A*X);
end
```

APPENDIX B: Part 2 Code

Adaptlc.m

```
%Set initial conditions
global count
global beta
global c

count=0;
count2 = 0;
beta= 0;
c = 0.1;
h=1;
hbig = 1;
hsmall = 1e-12;
hcut = 1.5;
hraise = 1.05;
tfin = 200;
epsilon=.0001;
t=0;
tmat = [];

x0 = 3.5;
y0 = 3.5;


%Improved Euler loop
while t<tfin %runs when h is sufficiently small

        %Go through IE Method while saving t value
        %temporary t
        if t+h>tfin
        h= tfin-t;
        end

        [dx,dy]= Prime(x0,y0);
        y1= y0 + h*dy;
        x1= x0+ h*dx;

        [dx1,dy1] = Prime(x1,y1);
        yh= y0+ (h/2)*(dy+dy1);
        xh= x0+ (h/2)*(y0+y1);

        %If difference between y1 and tempy divided by y1 is larger than
        %tolerance, h is adjusted until value is smaller than tolerance
        %calculates relative error
        k=abs(yh-y1);
        k2 = abs(xh-x1);
        if k2>=epsilon | k>=epsilon %if error is larger than our tolerance

        h = h/hcut;
        if h<hsmall %checks if new h is too small
        disp 'Step size is too small'%Give error message if h is too small
        %Set new values of t and yinit for next loop if there is no error
        h=hsmall*1.05;

        else
        end
        else
        t0=t;
        t=t+h;
        count2=count2+1;

        %Time, x(t) and y(t) values stored in matrix
        tmat(count2,1)=t;
        tmat(count2,2)= xh;
        tmat(count2,3)=yh;
        tmat(count2,4)=h;

        h= h*hraise;
        x0=xh;
        y0=yh;
        end
End
```

Prime.m

```
function [dx,dy]= Prime(x0,y0)
global beta
global count
global c

if abs(x0)<= 1
                f = -1;
                dy= -(x0)-c*f*y0;
                dx=y0;
elseif abs(x0)>1
                f = 1;
                dy= -(x0)-c*f*y0;
                dx=y0;
end

count=count+1;
end
```