# **Visual Improvements to the Phase-Plane Plot**

Hayden Atchison

EE 5323

10/12/2021

**Introduction** 1

**Symbolic Math** 1

**Jacobian Matrix** 3

**Syms and Jacobian Example without Additional Parameters** 4

**Equilibrium Points without Additional Parameters** 5

**Function Creation without Additional Parameters** 6

**Syms and Jacobian Example with Additional Parameters** 7

**Equilibrium Points with Additional Parameters** 8

**Function Creation with Additional Arguments** 9

**Quiver Plot and Annotations** 11

**Example 1: Homework 2, Question 3** 20

**Example 2: Visualization of Limit Cycles** 22

**Functions** 28

## **Introduction**

This started as a method for me to expand my knowledge for this course, but I ended up compiling this documentation of useful functions for graphics in this course. This may make the homework a bit too easy.

This section closes all open plots, clears the workspace and clears the command window, in addition to suppression of specific Editor warning messages.

close all; clear all; clc;%#ok<CLALL,\*DEFNU,\*SAGROW,\*NASGU,\*NOPTS>

## **Symbolic Math**

The Symbolic Math Toolbox lets you analytically perform:

* Differentiation
* Integration
* Simplification
* Transforms
* Equation solving
* And more

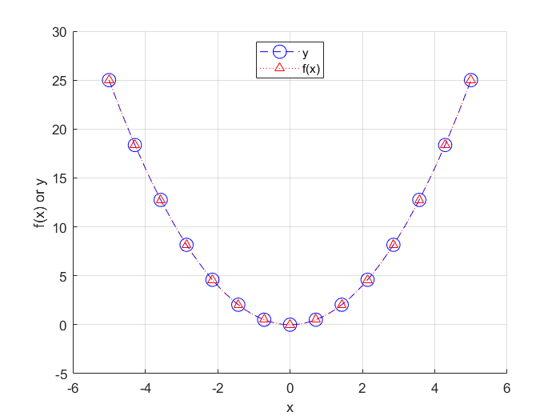
Functions can be defined as variables, or a combination such as:

syms x y f(x)  
y=x^2  
f(x)=x^2

y =  
   
x^2  
   
   
f(x) =  
   
x^2

To plot using symbolic variables, the function fplot is used. The variables x, f(x), and y don't need to be defined to plot, and as the figure below shows, the results from these two methods are identical.

figure('color','white');  
hold on  
grid  
fplot(x,y,'--ob','MarkerSize',10)  
fplot(x,f(x),':^r')  
xlabel('x')  
ylabel('f(x) or y')  
ylim([-5 30])  
xlim([-6 6])  
legend('y','f(x)','location','north')



We will be using the second method, as this lets us have  defined as

$$f(x,\dot{x})=\left[\matrix{x_1\cr x_2}\right]$$

This is represented in Matlab as:

syms f(x1,x2) x1 x2  
f(x1,x2)=[x1;x2]

f(x1, x2) =  
   
x1  
x2

## **Jacobian Matrix**

For calculation of the equilibrium points, we need to calculate the Jacobian. Luckily, the jacobian function supports symbolic math. The previous two methods, y=x and f(x)=x are shown below.

syms x y f(x)  
y=x^2-x  
J1=jacobian(y,x)

y =  
   
x^2 - x  
   
   
J1 =  
   
2\*x - 1

f(x)=x^2-x  
J2=jacobian(f(x),x)

f(x) =  
   
x^2 - x  
   
   
J2 =  
   
2\*x - 1

## **Syms and Jacobian Example without Additional Parameters**

This example uses the ODE from Homework 2, question 2. The equation is defined as:

$$f(x,y)=\left[\matrix{{y(1+x-y^2)}\cr {x(1+y-x^2)}}\right]$$

syms f(x,y) x y  
f(x,y)=[y\*(1+x-y^2);x\*(1+y-x^2)]  
J=jacobian(f(x,y),[x;y])

f(x, y) =  
   
y\*(- y^2 + x + 1)  
x\*(- x^2 + y + 1)  
   
   
J =  
   
[ y, - 3\*y^2 + x + 1]  
[- 3\*x^2 + y + 1, x]

## **Equilibrium Points without Additional Parameters**

In order to find the equilibrium points from the symbolic function, we need to solve the ODE for both x and y when f(x,y)=0. The solve function allows us to symbolically find the solution to this. Because we are solving for multiple variables, the output of the solve function is a structure of symbolic variables. Because there are no additional constants or parameters in these equations, we can directly convert the solution to a number using the double function.

F=solve(f==0,[x,y]);  
xe=[double(F.x),double(F.y)]

xe =  
  
 0 0  
 -1.0000 0  
 1.0000 0  
 0 -1.0000  
 0 1.0000  
 -0.6180 -0.6180  
 1.6180 1.6180

Even though the symbolic variables have been converted to numbers, we need to remove any imaginary equilibrium points from the results. This can be done by using boolean operations as array indices to see if they are imaginary, and for each of those, delete them with [].

xe(any(imag(xe),2),:)=[];

Because certain scenarios result in multiple equivalent equilibrium points, such as , we only save the unique values in our array of equilibrium points.

xe=unique(xe,'rows');  
x1e=xe(:,1);  
x2e=xe(:,2);  
fprintf('Equilibrium Points:\n\t x\t\ty\n')  
for i=1:numel(x1e)  
 fprintf('%6.4g %6.4g\n',x1e(i),x2e(i))  
end

Equilibrium Points:  
 x y  
 -1 0  
-0.618 -0.618  
 0 -1  
 0 0  
 0 1  
 1 0  
 1.618 1.618

## **Function Creation without Additional Parameters**

The function below is the one we will pass to ode23. Note that no additional parameters are passed to the function.

function out=D1(t,x)%#ok<INUSL>  
 out=[...  
 x(2)\*(1+x(1)-x(2)^2);...  
 x(1)\*(1+x(2)-x(1)^2);...  
 ];  
end

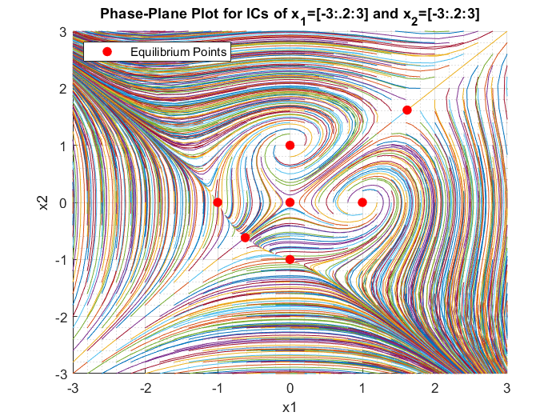
After we create our function, we can call it using ode23 or ode45. To make it easier to compensate for additional parameters, we will use a slightly modified input to the ode function. Based off of the way the function D1 above is coded, the output, out, is a structure comprised of x, representing the time vector, and y, representing one or more data vectors.

out=ode23(@(t,x) D1(t,x),tr,[x1 x2]);

The inputs to the ode function are different, with the handle @(t,x) representing what variables ode23 is looking for, followed by a space and the function ode23 will use in the solver. This format is necessary if you want to pass additional constants to the function we created, D1, without having multiple functions for each constant value.

Using the previously found equilibrium points, we can add them to the phase-plane plot. We need to create an empty plot that only contains the formatting we want on the legend, and then turn of legend updating so the hundreds of ode23 plots do not append to the legend. We then call ode23 with our function D1 in the for loops to create the phase-plane plot, followed by actually plotting the equilibrium points. This order is necessary so the equilibrium points show up over the ODE plots.

tr=[0 5];  
figure('color','white');  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-3:.2:3  
 for x2=-3:.2:3  
 out=ode23(@(t,x) D1(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title(sprintf('Phase-Plane Plot for ICs of x\_1=[-3:.2:3] and x\_2=[-3:.2:3]'))  
grid on  
grid minor  
xlim([-3 3])  
xlabel('x1')  
ylim([-3 3])  
ylabel('x2')



## **Syms and Jacobian Example with Additional Parameters**

This example uses the ODE from Homework 2, question 1. The equation is defined as:

$$f(x_1,x_2)=\left[\matrix{{x_2}\cr {-ax_1-x_1^3}}\right]$$

The symbolic math toolbox allows us to have the constant, a, in the equation, and also allows us to find the Jacobian without any issues.

syms f(x1,x2) x1 x2 a  
f(x1,x2)=[x2;-a\*x1-x1^3]  
J=jacobian(f(x1,x2),[x1;x2])

f(x1, x2) =  
   
 x2  
- x1^3 - a\*x1  
   
   
J =  
   
[ 0, 1]  
[- 3\*x1^2 - a, 0]

## **Equilibrium Points with Additional Parameters**

Similar to the last example, we solve f(x1,x2)=0, which gives us our symbolic equilibrium points.

F=solve(f==0,[x1,x2]);  
disp([F.x1,F.x2])

[ 0, 0]  
[ (-a)^(1/2), 0]  
[-(-a)^(1/2), 0]

To find the equilibrium points at various values of a, we need to substitute values of a using the subs function, which updates the symbolic expression with any constants that have been defined. The for loop defines four values of a, and these values are updated with subs. The equilibrium points and the number of equilibrium points are shown from the output of the code block below. It should be noted that we are saving each set of equilibrium points at the jth index of the xae variable as a cell array, since this lets us have multiple varying sized matrices of equilibrium points.

j=0;  
for a=[-1 -0.1 0 1]  
 j=j+1;  
 xe=[double(subs(F.x1)),double(subs(F.x2))];  
 fprintf('Unfiltered Equilibrium Points, a=%1.4g\n',a)  
 disp(xe)  
 xe(any(imag(xe),2),:)=[];  
 xe=unique(xe,'rows');  
 x1e=xe(:,1);  
 x2e=xe(:,2);  
 xae{j,1}=xe;  
 fprintf('For a=%1.4g, %0.0f Equilibrium Points:\n%6.4s %6.4s\n',a,numel(x1e),'x1','x2')  
 for i=1:numel(x1e)  
 fprintf('%6.3g %6.3g\n',x1e(i),x2e(i))  
 end  
 fprintf('\n')  
end

Unfiltered Equilibrium Points, a=-1  
 0 0  
 1 0  
 -1 0  
  
For a=-1, 3 Equilibrium Points:  
 x1 x2  
 -1 0  
 0 0  
 1 0  
  
Unfiltered Equilibrium Points, a=-0.1  
 0 0  
 0.3162 0  
 -0.3162 0  
  
For a=-0.1, 3 Equilibrium Points:  
 x1 x2  
-0.316 0  
 0 0  
 0.316 0  
  
Unfiltered Equilibrium Points, a=0  
 0 0  
 0 0  
 0 0  
  
For a=0, 1 Equilibrium Points:  
 x1 x2  
 0 0  
  
Unfiltered Equilibrium Points, a=1  
 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 1.0000i 0.0000 + 0.0000i  
 0.0000 - 1.0000i 0.0000 + 0.0000i  
  
For a=1, 1 Equilibrium Points:  
 x1 x2  
 0 0

## **Function Creation with Additional Arguments**

The function below is the one we will pass to ode23. Note that we have the constant a passed to the function.

function out=D2(t,x,a)%#ok<INUSL>  
 out=[...  
 x(2);...  
 -a\*x(1)-x(1)^3;...  
 ];  
end

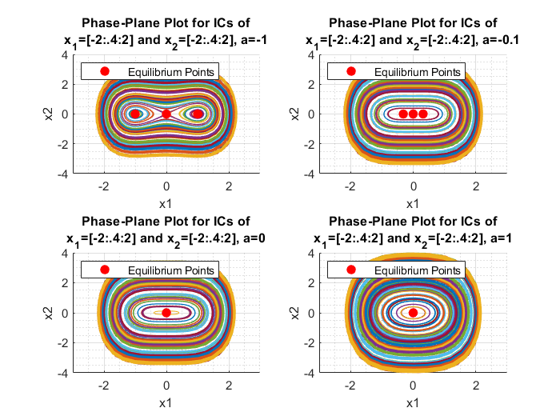
Similar to the previous example, nontraditional input arguments are used in the ode23 function.

out=ode23(@(t,x) D2(t,x,a),tr,[x1 x2]);

The additional input argument, a, only appears for the function we created, D2. This is because ode23 is only interested in the variables we gave the handle of @(t,x) to. This allows us to pass as many additional arguments to our custom function as we want.

Similar to the previous plot, we loop through the initial conditions  and , in addition to the constant a.

tr=[0 50];  
figure('color','white');  
i=0;  
for a=[-1 -0.1 0 1]  
 i=i+1;  
 subplot(2,2,i)  
 hold on  
 plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
 legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
 for x1=-2:.4:2  
 for x2=-2:.4:2  
 out=ode23(@(t,x) D2(t,x,a),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
 end  
 plot(xae{i,1}(:,1),xae{i,1}(:,2),'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
 title(sprintf('Phase-Plane Plot for ICs of\nx\_1=[-2:.4:2] and x\_2=[-2:.4:2], a=%1.4g',a))  
 grid on  
 grid minor  
 xlim([-3 3])  
 xlabel('x1')  
 ylim([-4 4])  
 ylabel('x2')  
 hold off  
end



## **Quiver Plot and Annotations**

For creation of the quiver plot, we will again use the equation from Homework 2, question 2. The section of code below finds the equilibrium points.

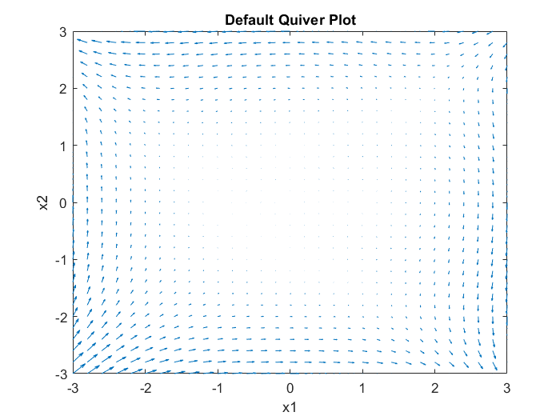
clear  
syms f(x,y) x y  
f(x,y)=[y\*(1+x-y^2);x\*(1+y-x^2)];  
J=jacobian(f(x,y),[x;y]);  
F=solve(f==0,[x,y]);  
xe=[double(F.x),double(F.y)];  
xe(any(imag(xe),2),:)=[];  
xe=unique(xe,'rows');  
x1e=xe(:,1);  
x2e=xe(:,2);  
fprintf('Equilibrium Points:\n\t x\t\ty\n')  
for i=1:numel(x1e)  
 fprintf('%6.4g %6.4g\n',x1e(i),x2e(i));  
end

Equilibrium Points:  
 x y  
 -1 0  
-0.618 -0.618  
 0 -1  
 0 0  
 0 1  
 1 0  
 1.618 1.618

For the creation of the quiver plot, a matrix of locations that vectors are assigned to must be created. I suggest using the same increment as the previous for loop from the phase-plane plot, or larger. The two arrays of points, x1 and x2, are "meshed" together. We then initialize the directional components of the vector, U and V. Using the previously created function D1, we run the function without ode23 to generate the vector for each initial condition pair, and then generate the quiver plot.

It should be noted that the handle to the quiver plot axis needs to be saved for use with the annotations.

figure('color','white');  
grid  
x1=-3:.2:3;  
x2=x1;  
[x1g,x2g]=meshgrid(x1,x2);  
U=zeros(size(x1g));  
V=U;  
t=0;  
for i=1:numel(x1g)-1  
 xp=D1(t,[x1g(i);x2g(i)]);  
 U(i)=xp(1);  
 V(i)=xp(2);  
end  
qaxis=quiver(x1g,x2g,U,V);  
title('Default Quiver Plot')  
xlim([-3 3])  
xlabel('x1')  
ylim([-3 3])  
ylabel('x2')

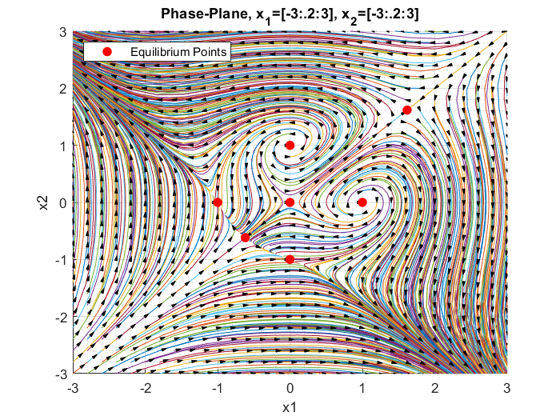


To add the annotations, we extract the data from the quiver axis handle we created, qaxis.

U = qaxis.UData;  
V = qaxis.VData;  
X = qaxis.XData;  
Y = qaxis.YData;

We then prepare a for loop as if we are creating a phase-plane plot, as we want this to be the first plot so the arrow annotations are placed over it. The arrow line length must be set small, because the arrow tip points to the tip of the vector, and the arrows will be misaligned in the annotation. A two-level for loop is created to place an arrow at each location, similar to the loop structure required for the quiver plot. We then finalize by overlaying the equilibrium points found previously.

headWidth = 2;  
headLength = 4;  
LineLength = 0.005;  
tr=[0 5];  
figure('color','white');  
grid  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-3:.2:3  
 for x2=-3:.2:3  
 out=ode23(@(t,x) D1(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
for ii=1:length(X)  
 for ij=1:length(X)  
 ah=annotation('arrow','headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title('Phase-Plane, x\_1=[-3:.2:3], x\_2=[-3:.2:3]')  
grid on  
grid minor  
xlim([-3 3])  
xlabel('x1')  
ylim([-3 3])  
ylabel('x2')



For those of you who are more "enthusiastic" about Matlab and want to make the plot even better, you can additionally scale the arrow head width and length based on the value of the vector from the quiver plot, but that will not be covered here.

The section below uses the function with varying values of a to show what the quiver plots look like, and to verify the equilibrium points.

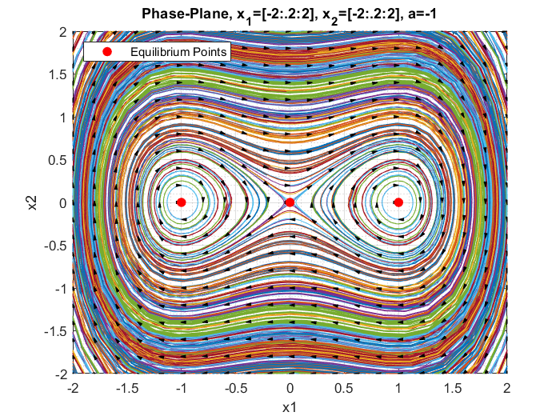
clear  
syms f(x1,x2) x1 x2 a  
f(x1,x2)=[x2;-a\*x1-x1^3]  
J=jacobian(f(x1,x2),[x1;x2])  
F=solve(f==0,[x1,x2]);  
disp([F.x1,F.x2])

f(x1, x2) =  
   
 x2  
- x1^3 - a\*x1  
   
   
J =  
   
[ 0, 1]  
[- 3\*x1^2 - a, 0]  
   
[ 0, 0]  
[ (-a)^(1/2), 0]  
[-(-a)^(1/2), 0]

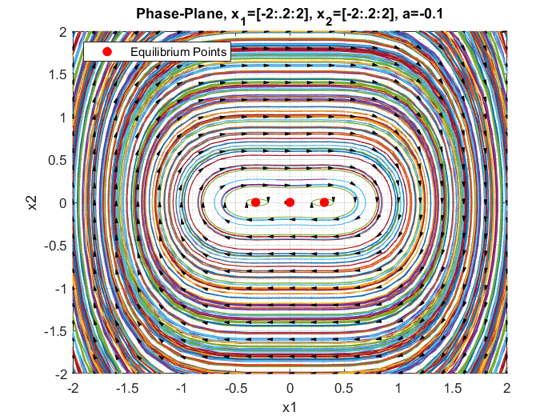
j=0;  
for a=[-1 -0.1 0 1]

figure('color','white');  
 j=j+1;  
 xe=[double(subs(F.x1)),double(subs(F.x2))];  
 fprintf('Unfiltered Equilibrium Points, a=%1.4g\n',a)  
 disp(xe)  
 xe(any(imag(xe),2),:)=[];  
 xe=unique(xe,'rows');  
 x1e=xe(:,1);  
 x2e=xe(:,2);  
 xae{j,1}=xe;  
 fprintf('For a=%1.4g, %0.0f Equilibrium Points:\n%6.4s %6.4s\n',a,numel(x1e),'x1','x2')  
 for i=1:numel(x1e)  
 fprintf('%6.3g %6.3g\n',x1e(i),x2e(i))  
 end  
 fprintf('\n')  
 grid  
 x1=-2:.2:2;  
 x2=x1;  
 [x1g,x2g]=meshgrid(x1,x2);  
 U=zeros(size(x1g));  
 V=U;  
 t=0;  
 for i=1:numel(x1g)-1  
 xp=D2(t,[x1g(i);x2g(i)],a);  
 U(i)=xp(1);  
 V(i)=xp(2);  
 end  
 qaxis=quiver(x1g,x2g,U,V);  
 xlim([-2 2])  
 ylim([-2 2])  
 U = qaxis.UData;  
 V = qaxis.VData;  
 X = qaxis.XData;  
 Y = qaxis.YData;  
 headWidth = 2;  
 headLength = 4;  
 LineLength = 0.005;  
 tr=[0 10];  
 cla(gca)  
 hold on  
 plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
 legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
 for x1=-2:.2:2  
 for x2=-2:.2:2  
 out=ode23(@(t,x) D2(t,x,a),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
 end  
 for ii = 1:length(X)  
 for ij = 1:length(X)  
 ah = annotation('arrow',...  
 'headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
 end  
 plot(xae{j,1}(:,1),xae{j,1}(:,2),'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
 title(sprintf('Phase-Plane, x\_1=[-2:.2:2], x\_2=[-2:.2:2], a=%1.4g',a))  
 grid on  
 grid minor  
 xlim([-2 2])  
 xlabel('x1')  
 ylim([-2 2])  
 ylabel('x2')  
 hold off

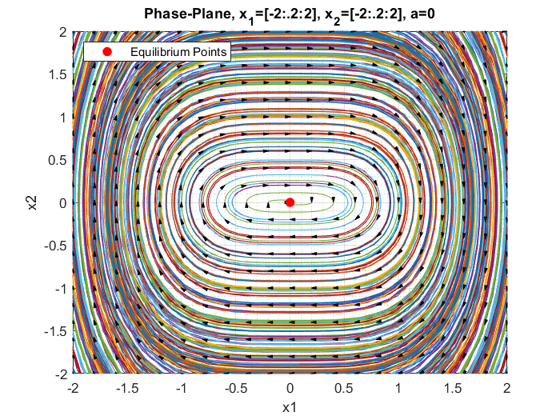
Unfiltered Equilibrium Points, a=-1  
 0 0  
 1 0  
 -1 0  
  
For a=-1, 3 Equilibrium Points:  
 x1 x2  
 -1 0  
 0 0  
 1 0



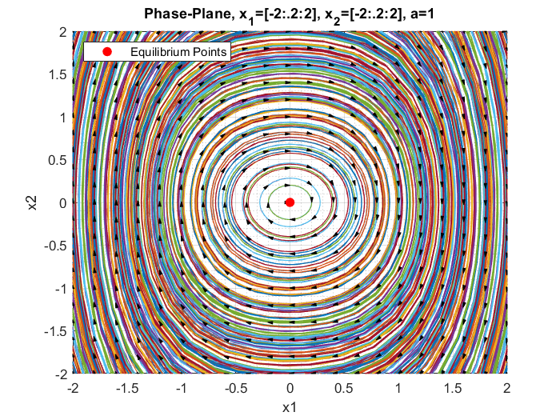
Unfiltered Equilibrium Points, a=-0.1  
 0 0  
 0.3162 0  
 -0.3162 0  
  
For a=-0.1, 3 Equilibrium Points:  
 x1 x2  
-0.316 0  
 0 0  
 0.316 0



Unfiltered Equilibrium Points, a=0  
 0 0  
 0 0  
 0 0  
  
For a=0, 1 Equilibrium Points:  
 x1 x2  
 0 0



Unfiltered Equilibrium Points, a=1  
 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 1.0000i 0.0000 + 0.0000i  
 0.0000 - 1.0000i 0.0000 + 0.0000i  
  
For a=1, 1 Equilibrium Points:  
 x1 x2  
 0 0



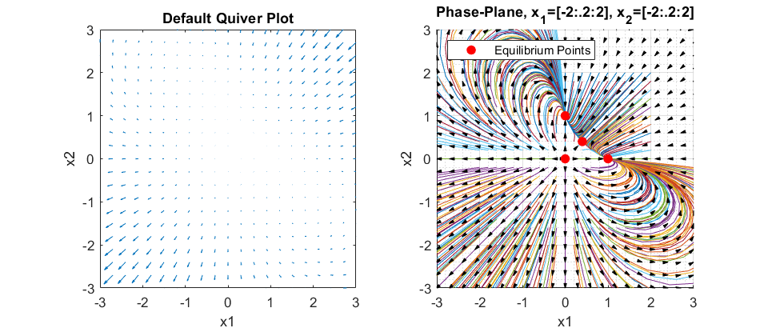
end

## **Example 1: Homework 2, Question 3**

Example using the equation from Homework 2, question 3:

$$f(x1,x2)=\left[\matrix{{2x_1-3x_1x_2-2x_1^2}\cr {2x_2-3x_1x_2-2x_2^2}}\right]$$

clear  
syms f(x1,x2) x1 x2  
f(x1,x2)=[2\*x1-3\*x1\*x2-2\*x1^2;2\*x2-3\*x1\*x2-2\*x2^2];  
J=jacobian(f(x1,x2),[x1;x2]);  
F=solve(f==0,[x1,x2]);  
x1e=double(F.x1);  
x2e=double(F.x2);  
figure('color','white');  
subplot(1,2,1)  
grid  
x1=-3:.3:3;  
x2=x1;  
[x1g,x2g]=meshgrid(x1,x2);  
U=zeros(size(x1g));  
V=U;  
t=0;  
for i=1:numel(x1g)-1  
 xp=D3(t,[x1g(i);x2g(i)]);  
 U(i)=xp(1);  
 V(i)=xp(2);  
end  
qaxis=quiver(x1g,x2g,U,V);  
title('Default Quiver Plot')  
xlim([-3 3])  
xlabel('x1')  
ylim([-3 3])  
ylabel('x2')  
U = qaxis.UData;  
V = qaxis.VData;  
X = qaxis.XData;  
Y = qaxis.YData;  
headWidth = 2;  
headLength = 4;  
LineLength = 0.0005;  
tr=[0 5];  
subplot(1,2,2)  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-2:.2:2  
 for x2=-2:.2:2  
 out=ode23(@(t,x) D3(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
for ii = 1:length(X)  
 for ij = 1:length(X)  
 ah = annotation('arrow',...  
 'headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title('Phase-Plane, x\_1=[-2:.2:2], x\_2=[-2:.2:2]')  
grid on  
grid minor  
xlim([-3 3])  
xlabel('x1')  
ylim([-3 3])  
ylabel('x2')  
cf=gcf;  
cf.Position=[1,454,766,330];



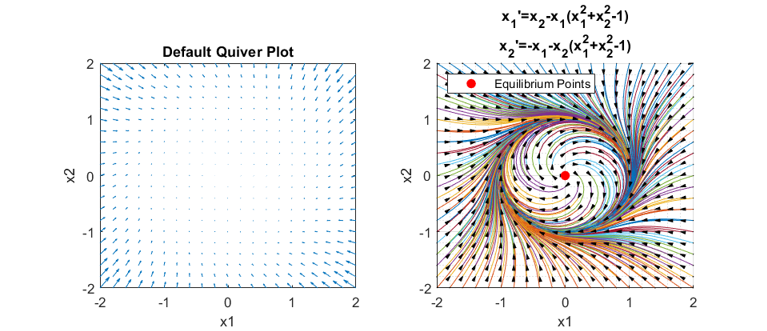
## **Example 2: Visualization of Limit Cycles**

The annotated phase-plane plot can also be used for easy visualization of limit cycles. This example will go over the three parts of Example 2.7 in the book. For the first equation, equilibrium points outside the limit cycle radius of 1 converge to the limit cycle, and equilibrium points inside the limit cycle diverge from the center equilibrium point towards the limit cycle.

The equation for the first part is as follows:

$$f(x1,x2)=\left[\matrix{{x_2-x_1(x_1^2+x_2^2-1)}\cr {-x_1-x_2(x_1^2+x_2^2-1)}}\right]$$

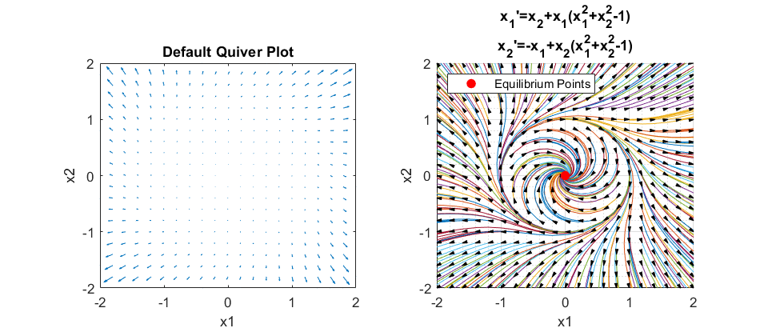
clear  
syms f(x1,x2) x1 x2  
f(x1,x2)=[x2-x1\*(x1^2+x2^2-1);-x1-x2\*(x1^2+x2^2-1)];  
J=jacobian(f(x1,x2),[x1;x2]);  
F=solve(f==0,[x1,x2]);  
x1e=double(F.x1);  
x2e=double(F.x2);  
figure('color','white');  
subplot(1,2,1)  
grid  
x1=-2:.2:2;  
x2=x1;  
[x1g,x2g]=meshgrid(x1,x2);  
U=zeros(size(x1g));  
V=U;  
t=0;  
for i=1:numel(x1g)-1  
 xp=D4(t,[x1g(i);x2g(i)]);  
 U(i)=xp(1);  
 V(i)=xp(2);  
end  
qaxis=quiver(x1g,x2g,U,V);  
title('Default Quiver Plot')  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
U = qaxis.UData;  
V = qaxis.VData;  
X = qaxis.XData;  
Y = qaxis.YData;  
headWidth = 2;  
headLength = 4;  
LineLength = 0.001;  
tr=[0 10];  
subplot(1,2,2)  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-2:.2:2  
 for x2=-2:.2:2  
 out=ode23(@(t,x) D4(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
for ii = 1:length(X)  
 for ij = 1:length(X)  
 ah = annotation('arrow',...  
 'headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title(sprintf('x\_1''=x\_2-x\_1(x\_1^2+x\_2^2-1)\nx\_2''=-x\_1-x\_2(x\_1^2+x\_2^2-1)'))  
grid on  
grid minor  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
cf=gcf;  
cf.Position=[1,454,766,330];



For this ODE, the equilibrium points outside the limit cycle radius of 1 diverge from the limit cycle, and equilibrium points inside the limit cycle converge to the center equilibrium point away from the limit cycle.

$$f(x1,x2)=\left[\matrix{{x_2+x_1(x_1^2+x_2^2-1)}\cr {-x_1+x_2(x_1^2+x_2^2-1)}}\right]$$

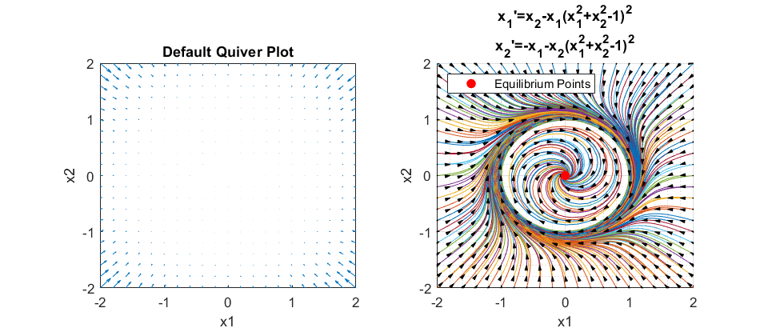
clear  
syms f(x1,x2) x1 x2  
f(x1,x2)=[x2+x1\*(x1^2+x2^2-1);-x1+x2\*(x1^2+x2^2-1)];  
J=jacobian(f(x1,x2),[x1;x2]);  
F=solve(f==0,[x1,x2]);  
x1e=double(F.x1);  
x2e=double(F.x2);  
figure('color','white');  
subplot(1,2,1)  
grid  
x1=-2:.2:2;  
x2=x1;  
[x1g,x2g]=meshgrid(x1,x2);  
U=zeros(size(x1g));  
V=U;  
t=0;  
for i=1:numel(x1g)-1  
 xp=D5(t,[x1g(i);x2g(i)]);  
 U(i)=xp(1);  
 V(i)=xp(2);  
end  
qaxis=quiver(x1g,x2g,U,V);  
title('Default Quiver Plot')  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
U = qaxis.UData;  
V = qaxis.VData;  
X = qaxis.XData;  
Y = qaxis.YData;  
headWidth = 2;  
headLength = 4;  
LineLength = 0.001;  
tr=[0 10];  
subplot(1,2,2)  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-2:.2:2  
 for x2=-2:.2:2  
 out=ode23(@(t,x) D5(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
for ii = 1:length(X)  
 for ij = 1:length(X)  
 ah = annotation('arrow',...  
 'headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title(sprintf('x\_1''=x\_2+x\_1(x\_1^2+x\_2^2-1)\nx\_2''=-x\_1+x\_2(x\_1^2+x\_2^2-1)'))  
grid on  
grid minor  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
cf=gcf;  
cf.Position=[1,454,766,330];



For this ODE, the equilibrium points outside the limit cycle radius of 1 converge towards the limit cycle, and equilibrium points inside the limit cycle converge to the center equilibrium point away from the limit cycle.

$$f(x1,x2)=\left[\matrix{{x_2-x_1(x_1^2+x_2^2-1)^2}\cr {-x_1-x_2(x_1^2+x_2^2-1)^2}}\right]$$

clear  
syms f(x1,x2) x1 x2  
f(x1,x2)=[x2-x1\*(x1^2+x2^2-1)^2;-x1-x2\*(x1^2+x2^2-1)^2];  
J=jacobian(f(x1,x2),[x1;x2]);  
F=solve(f==0,[x1,x2]);  
x1e=double(F.x1);  
x2e=double(F.x2);  
figure('color','white');  
subplot(1,2,1)  
grid  
x1=-2:.2:2;  
x2=x1;  
[x1g,x2g]=meshgrid(x1,x2);  
U=zeros(size(x1g));  
V=U;  
t=0;  
for i=1:numel(x1g)-1  
 xp=D6(t,[x1g(i);x2g(i)]);  
 U(i)=xp(1);  
 V(i)=xp(2);  
end  
qaxis=quiver(x1g,x2g,U,V);  
title('Default Quiver Plot')  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
U = qaxis.UData;  
V = qaxis.VData;  
X = qaxis.XData;  
Y = qaxis.YData;  
headWidth = 2;  
headLength = 4;  
LineLength = 0.001;  
tr=[0 10];  
subplot(1,2,2)  
hold on  
plot(nan,nan,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
legend('Equilibrium Points','location','northwest','AutoUpdate','off')  
for x1=-2:.2:2  
 for x2=-2:.2:2  
 out=ode23(@(t,x) D6(t,x),tr,[x1 x2]);  
 plot(out.y(1,:),out.y(2,:))  
 end  
end  
for ii = 1:length(X)  
 for ij = 1:length(X)  
 ah = annotation('arrow',...  
 'headStyle','cback1','HeadLength',headLength,'HeadWidth',headWidth,'LineStyle','none');  
 set(ah,'parent',gca);  
 set(ah,'position',[X(ii,ij) Y(ii,ij) LineLength\*U(ii,ij) LineLength\*V(ii,ij)]);  
 end  
end  
plot(x1e,x2e,'o','linewidth',1,'markeredgecolor','r','markerfacecolor','r')  
title(sprintf('x\_1''=x\_2-x\_1(x\_1^2+x\_2^2-1)^2\nx\_2''=-x\_1-x\_2(x\_1^2+x\_2^2-1)^2'))  
grid on  
grid minor  
xlim([-2 2])  
xlabel('x1')  
ylim([-2 2])  
ylabel('x2')  
cf=gcf;  
cf.Position=[1,454,766,330];



## **Functions**

function out=D1Ex(t,x)%#ok<INUSL>  
 out=[...  
 x(2)\*(1+x(1)-x(2)^2);...  
 x(1)\*(1+x(2)-x(1)^2);...  
 ];  
end  
function out=D2Ex(t,x,a)%#ok<INUSL>  
out=[...  
 x(2);...  
 -a\*x(1)-x(1)^3;...  
 ];  
end  
function out=D3Ex(t,x)%#ok<INUSL>  
out=[...  
 2\*x(1)-3\*x(1)\*x(2)-2\*x(1)^2;...  
 2\*x(2)-3\*x(1)\*x(2)-2\*x(2)^2;...  
 ];  
end  
function out=D4Ex(t,x)%#ok<INUSL>  
out=[...  
 x(2)-x(1)\*(x(1)^2+x(2)^2-1);...  
 -x(1)-x(2)\*(x(1)^2+x(2)^2-1);...  
 ];  
end  
function out=D5Ex(t,x)%#ok<INUSL>  
out=[...  
 x(2)+x(1)\*(x(1)^2+x(2)^2-1);...  
 -x(1)+x(2)\*(x(1)^2+x(2)^2-1);...  
 ];  
end  
function out=D6Ex(t,x)%#ok<INUSL>  
out=[...  
 x(2)-x(1)\*(x(1)^2+x(2)^2-1)^2;...  
 -x(1)-x(2)\*(x(1)^2+x(2)^2-1)^2;...  
 ];  
end

[*Published with MATLAB® R2020b*](https://www.mathworks.com/products/matlab)