Chapter 9

Nonlinear Systems and Phenomena

Application 9.1
Phase Plane Portraits and First-Order Equations

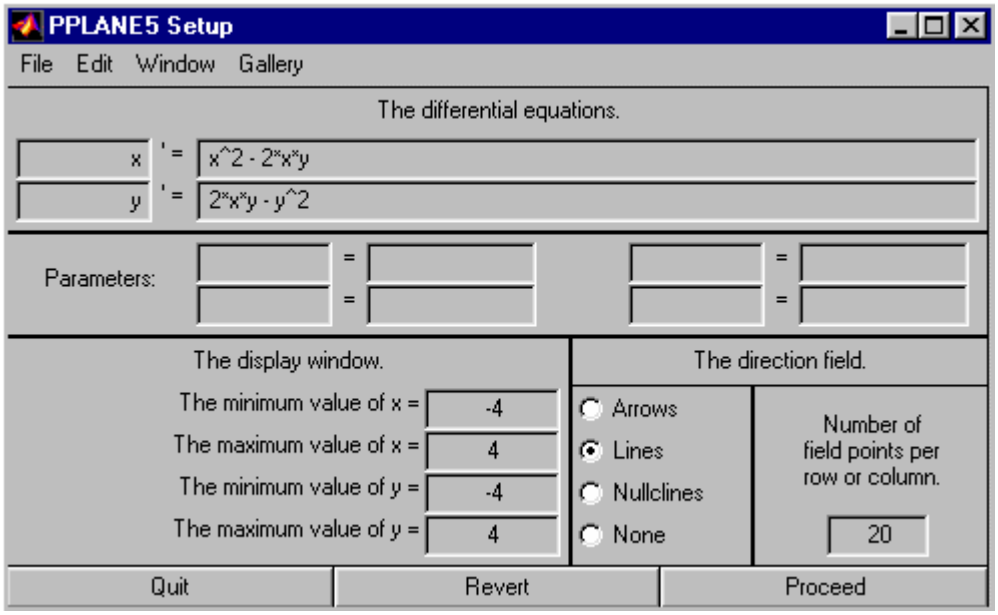
Consider a first-order differential equation of the form

$$\frac{dy}{dx} = \frac{G(x, y)}{F(x, y)}, \tag{1}$$

which may be difficult or impossible to solve explicitly. Its solution curves can nevertheless be plotted as trajectories of the corresponding autonomous two-dimensional system

$$\frac{dx}{dt} = F(x, y), \qquad \frac{dy}{dt} = G(x, y). \tag{2}$$

Most ODE plotters can routinely generate phase portraits for autonomous systems. Many of those appearing in Chapter 9 of the text were plotted using (as illustrated in the figure below) John Polking's **pplane** program (see J. Polking and D. Arnold, *Ordinary Differential Equations Using MATLAB* (2nd edition), Prentice Hall, 1999).



For example, to plot solution curves for the differential equation

$$\frac{dy}{dx} = \frac{2xy - y^2}{x^2 - 2xy} \tag{3}$$

we plot trajectories of the system

$$\frac{dx}{dt} = x^2 - 2xy\,, \qquad \frac{dy}{dt} = 2xy - y^2\,. \tag{4}$$

with the **`pplane`** setup of the preceding figure. The result is shown in Fig. 9.1.20 of the text. In the sections that follow the problems below we discuss the use of *Maple*, *Mathematica*, and MATLAB to construct such phase plane portraits.

Plot similarly some solution curves for the following differential equations.

1. $\dfrac{dy}{dx} = \dfrac{4x - 5y}{2x + 3y}$

2. $\dfrac{dy}{dx} = \dfrac{4x - 5y}{2x - 3y}$

3. $\dfrac{dy}{dx} = \dfrac{4x - 3y}{2x - 5y}$

4. $\dfrac{dy}{dx} = \dfrac{2xy}{x^2 - y^2}$

5. $\dfrac{dy}{dx} = \dfrac{x^2 + 2xy}{y^2 + 2xy}$

Now construct some examples of your own. Homogeneous functions like those in Problems 1 through 5 — rational functions with numerator and denominator of the same degree in x and y — work well. The differential equation

$$\frac{dy}{dx} = \frac{25x + y(1 - x^2 - y^2)(4 - x^2 - y^2)}{-25y + x(1 - x^2 - y^2)(4 - x^2 - y^2)} \tag{5}$$

of this form generalizes Example 6 in Section 9.1 of the text, but would be inconvenient to solve explicitly. Its phase portrait (Fig. 9.1.21) shows two periodic closed trajectories — the circles $r = 1$ and $r = 2$. Anyone want to try for three circles?

Using *Maple*

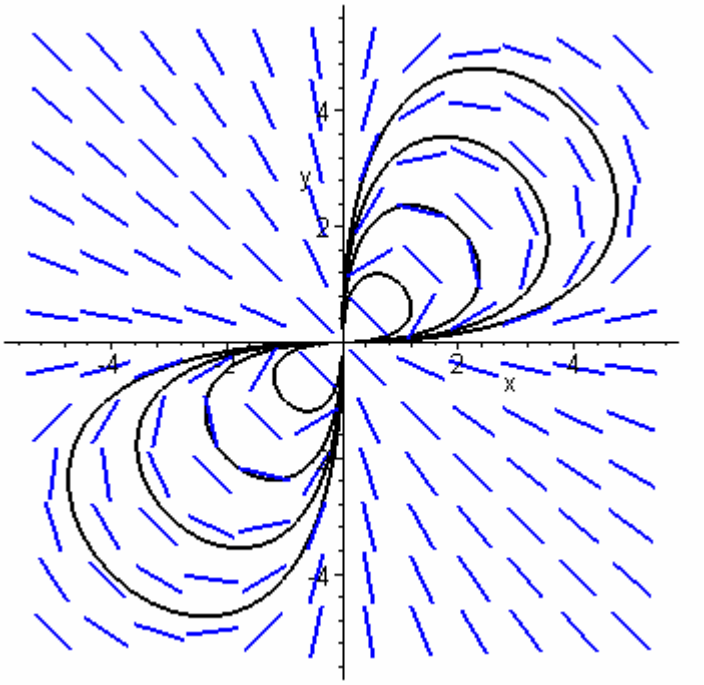
The **DEtools** package includes the **DEplot** function that can be used to construct a phase plane portrait for a 2-dimensional system of first-order differential equations. For instance, given the differential equations

```
deq1 := diff(x(t),t) = x^2 - 2*x*y:
deq2 := diff(y(t),t) = 2*x*y - y^2:
```

in (4), in order to plot the trajectories with initial conditions $x(0) = n$, $y(0) = n$ for $n = -4, -3, ..., 2, 3, 4,$ we need only enter the commands **with(DEtools):** and

```
DEplot([deq1,deq2], [x,y],
    t=-10..10, x=-5..5, y=-5..5,
   {[x(0)=-4,y(0)=-4],[x(0)=-3,y(0)=-3],
    [x(0)=-2,y(0)=-2],[x(0)=-1,y(0)=-1],
    [x(0)=1,y(0)=1],[x(0)=2,y(0)=2],
    [x(0)=3,y(0)=3],[x(0)=4,y(0)=4]},
    arrows = line, stepsize=0.02);
```

specifying the differential equations, the dependent variables, the t-range for each solution curve and the xy-plot ranges, a list of initial conditions for the desired solution curves, and the desired step size using the default Runge-Kutta method. The result is the collection of first- and third-quadrant trajectories shown in the figure below. You can investigate similarly the second- and fourth-quadrant trajectories of the system in (4).



Using *Mathematica*

First we define the differential equations

```
deq1 = x'[t] == x[t]^2 - 2*x[t]*y[t];
deq2 = y'[t] == 2*x[t]*y[t] -  y[t]^2;
```

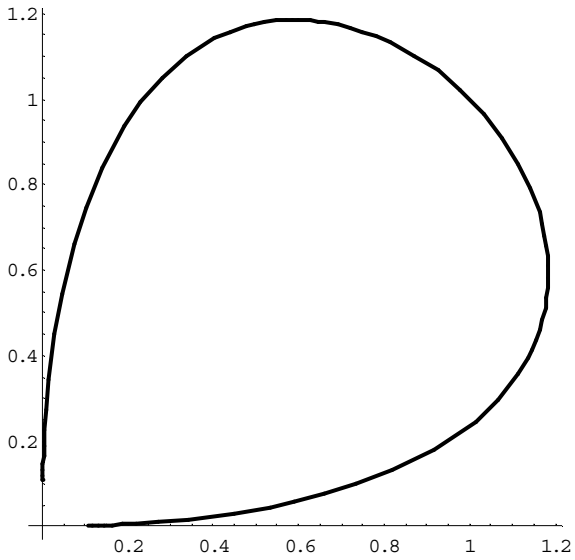
in (4). Then we can use the **NDSolve** function to solve the system numerically with given inital conditions.

```
soln =
NDSolve[ {deq1, deq2, x[0]==1, y[0]==1},
          {x[t],y[t]}, {t,-10,10} ]

{{x[t] -> InterpolatingFunction[{-10., 10.}, <>][t],
   y[t] -> InterpolatingFunction[{-10., 10.}, <>][t]}}
```

The corresponding solution curve is plotted with the commands

```
x = soln[[1,1,2]];
y = soln[[1,2,2]];
curve =
ParametricPlot[ {x,y}, {t,-10,10} ];
```



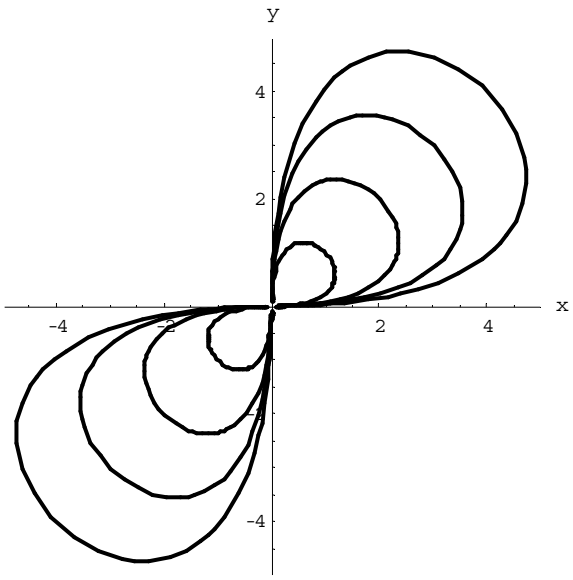
We can plot as many such solution curves as we like, and then display them simultaneously. For example, the following command creates a list (a **Table**) of solution curves corresponding to the initial conditions $x(0) = n$, $y(0) = n$ for $n = -4, -3, \ldots, 2, 3, 4$.

```
curve = Table[ n, {n,-4,4} ];
Do[Clear[x,y];
   soln =
   NDSolve[ {deq1,deq2,x[0] == n,y[0]== n},
            {x[t],y[t]}, {t,-10,10} ];
   x = soln[[1,1,2]];
   y = soln[[1,2,2]];
   curve[[n]] =
   ParametricPlot[ {x,y},
            {t,-10,10} ], {n,-4,4} ];
```

The command

```
Show[curve[[-4]],curve[[-3]],curve[[-2]],curve[[-1]],
     curve[[1]], curve[[2]], curve[[3]], curve[[4]],
     AspectRatio -> 1 ];
```

then displays these solution curves in a single phase plane portrait.



You can investigate similarly the second- and fourth-quadrant trajectories of the system in (4).

Using MATLAB

We will use the MATLAB ODE-solver **ode23** to construct a phase plane portrait for the system in (4). First we save the MATLAB m-file **yp.m** consisting of the lines

```
function    yp = yp(t,y)
%  yp.m
yp = y;
x = y(1);
y = y(2);
yp(1) = x.*x - 2*x.*y;
yp(2) = 2*x.*y - y.*y;
```

to define the system. Then the command

```
[t,y] = ode23('yp', [0,10], [4;4]);
```

generates a 2-column matrix **y** whose two columns are the x- and y-vectors for an approximate solution of the system on the interval $0 \le t \le 10$ with initial values $x(0) = y(0) = 4$. However, we may be surprised to find that the plot command

```
plot( y(:,1), y(:,2) )
```

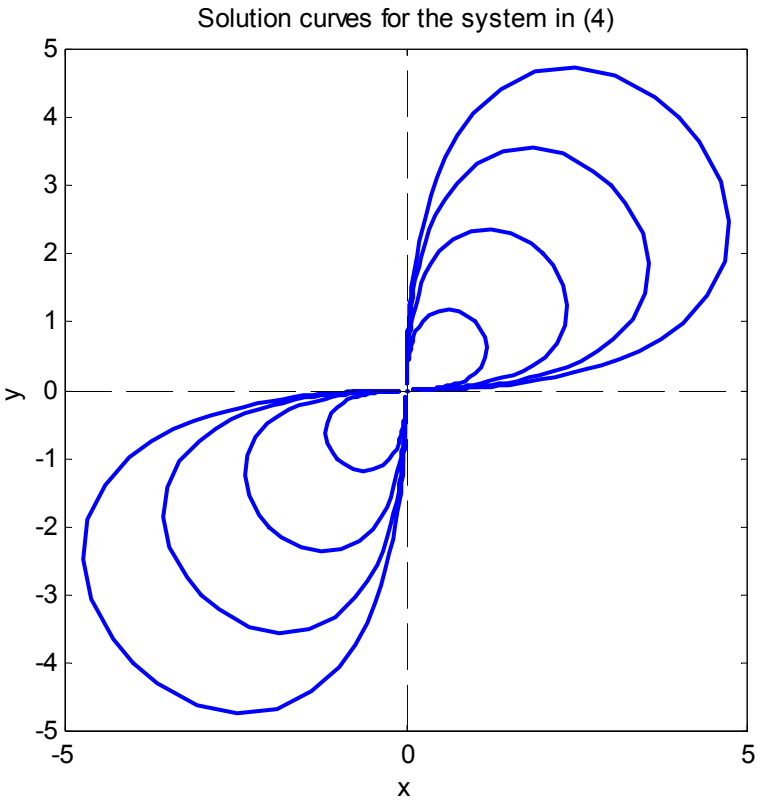
then produces only half of the expected first-quadrant loop trajectory. (Try it and see for yourself!)

The other half of the desired "whole" trajectory is obtained by solving the solution system in (4) with the same initial condition, but on the interval $-10 \le t \le 0$. Actually, the whole loop corresponds to the infinite parameter interval $-\infty < t < \infty$ with the point $(x(t), y(t))$ on the trajectory approaching the origin as $t \to \pm\infty$, but the finite interval $-10 < t < 10$ suffices for graphical purposes.

We therefore proceed to generate directly a family of "whole" trajectories, solving the system numerically with the initial conditions $x(0) = y(0) = n$, twice for each of the successive values $n = -4, -3, \ldots, 2, 3, 4$. This is done by the simple **for**-loop

```
for n = -4 : 4
    [t,y] = ode23('yp', [0,10], [n;n] );
    plot(y(:,1),y(:,2),'b')
    hold on
    [t,y] = ode23('yp', [0,-10], [n;n] );
    plot(y(:,1),y(:,2),'b')
    end
```

which plots successively both "halves" of each first- and third-quadrant trajectory. The result is the figure shown on the next page. You can alter the initial conditions and t-intervals specified to investigate similarly the second- and fourth-quadrant trajectories of the system in (4).


Solution curves for the system in (4)
5
4
3
2
1
0
-1
-2
-3
-4
-5
y
-5
0
5
x