22

Differential and Difference Equations

Introduction 651

22.1 Symbolic Solutions 652

22.1.1 First-Order Equations 652 **DSolve**
22.1.2 Second- and Higher-Order Equations 656
22.1.3 Simultaneous Equations 660

22.2 More about Symbolic Solutions 663

22.2.1 Calculating with the Solution 663
22.2.2 Using the Laplace Transform 666 **LaplaceTransform**
22.2.3 Series Solutions 668 **LogicalExpand**

22.3 Numerical Solutions 669

22.3.1 One Equation 669 **NDSolve**
22.3.2 Two Equations 674
22.3.3 Three Equations 680

22.4 More about Numerical Solutions 684

22.4.1 Options 684
22.4.2 Runge–Kutta Method 687 **rk**
22.4.3 Boundary Value Problems 689 **linbvp**, **nonlinbvp**

22.5 Difference Equations 692

22.5.1 Linear Difference Equations 692 **RSolve**, **GeneratingFunction**, **SeriesTerm**
22.5.2 Nonlinear Difference Equations 694 **fixedPlot**
22.5.3 Bifurcation and Lyapunov Exponent 697 **bifurcation**, **lyapunov**

Introduction

God is not so cruel as to create situations described by nonlinear differential equations. — Edward Sexton

Solving ordinary differential equations with *Mathematica* is straightforward: we have **DSolve** for symbolic solution and **NDSolve** for numerical solution. Both commands accept one or more equations, first- or higher-order equations, linear and nonlinear equations, and solve

both initial and boundary value problems (**NDSolve** only solves linear boundary value problems).

In addition we consider solving differential equations with the Laplace transform and finding series solutions. We implement the Runge–Kutta method and also some methods for boundary value problems.

At the end of this chapter we consider difference equations. If such an equation is linear, a solution in closed form can possibly be found. Nonlinear difference equations cannot generally be solved but they can be analyzed otherwise with *Mathematica*. We plot, for example, a bifurcation diagram and a figure for the Lyapunov exponent.

For more about differential equations with *Mathematica*, see, for example, Abell & Braselton (1997).

22.1 Symbolic Solutions

22.1.1 First-Order Equations

Here are some common commands for first-order differential equations.

`DSolve[eq, y[x], x]`	Give the general solution
`sol = DSolve[{eq, y[x0]==y0}, y[x], x]`	Give the solution of the initial value problem
`Plot[Evaluate[y[x]/.sol], {x,a,b}]`	Plot the solution of the initial value problem

An example of a differential equation is `y'[x] == a y[x] + b x + c`. The dependent variable, here `y`, must contain the independent variable, here `x`, as the argument, that is, we cannot write the equation as `y' == a y + b x + c`. Note, too, that the equation must contain `==` (not `=`) and the initial condition must also contain `==` (not `=`).

■ Example 1: General Solution

Here is a first-order nonlinear equation:

```
eq = x'[t] == k(xf - x[t])x[t];
```

Note that now `t` is the independent variable and `x` the dependent variable; `k` and `xf` are constants. It may be convenient to give a name for the equation here (you can also enter the whole equation in **DSolve**). It is often convenient to give a name for the solution, too, for easy reference:

```
sol = DSolve[eq, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{E^{k\,t\,xf}\,xf}{E^{k\,t\,xf} - E^{C[1]}}\right\}\right\}$$

The solution is given in the form of a rule (for more about rules, see Section 12.1.2). The arbitrary constant is **C[1]** (we can give it whatever value we want).

In general, the solution given by **DSolve** consists of a list of solutions:

```
{{solution 1}, {solution 2}, ...}
```

because a given equation can have several solutions. Each solution is again a list consisting of as many elements as there are dependent variables. In our example we have only one dependent variable, **x**, and we got only one solution. Thus the solution is of the form `{{solution 1 for x}}`.

Note that the solution **sol** is a generic solution, that is, a solution valid for general values of the parameters **k** and **xf**. For some particular values the solution may be of a different form. For example, when **xf** is zero, the solution is

```
DSolve[eq/.xf->0, x[t], t]
```

$$\{\{x[t] \to \frac{1}{k\,t - C[1]}\}\}$$

■ Example 2: Initial Value Problem

Next we solve an initial value problem:

```
DSolve[{eq, x[0]==x0}, x[t], t] //Simplify
```

$$\{\{x[t] \to \frac{E^{k\,t\,xf}\,x0\,xf}{(-1 + E^{k\,t\,xf})\,x0 + xf}\}\}$$

Now we give specific values for all constants:

```
sol = DSolve[{eq /. {k->1/10, xf->10}, x[0]==1/2}, x[t], t]
```

$$\{\{x[t] \to \frac{10\,E^{t}}{19 + E^{t}}\}\}$$

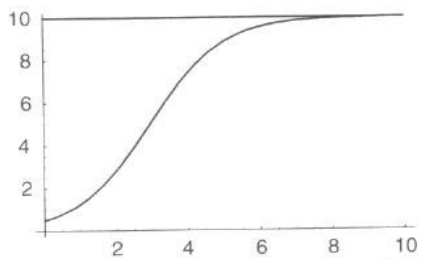
This solution can be plotted, because it does not contain any parameters. To get the expression of the solution we can write

```
x[t] /. sol
```

$$\{\frac{10\,E^{t}}{19 + E^{t}}\}$$

so that to plot the solution we write

```
g1 = Plot[{10, Evaluate[x[t]/.sol]}, {t,0,10}];
```



This is a so-called logistic curve. We also plotted the asymptote 10 of this curve. Here is a table of values:

```
Table[{t, x[t]/.sol}, {t,0.,5,1}] //TableForm //PaddedForm
```

```
0.      0.5
1.      1.25161
2.      2.80005
3.      5.13887
4.      7.41841
5.      8.86508
```

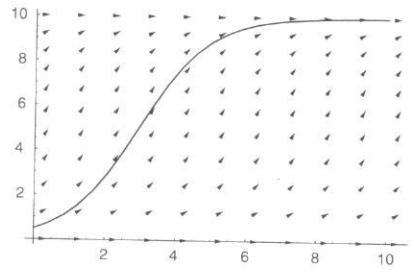
■ Example 3: Direction Field

We can learn the behavior of the solutions of a differential equation by plotting a set of arrows that are tangent to the solution. The plot is called a direction field. It can be plotted with **PlotVectorField** from a package (see Section 7.2.5). This command plots vectors **{exprx,expry}** for some values of **x** and **y**. We can choose **exprx** to be **1** and **expry** to be **y'[x]**. So we write:

```
Needs["Graphics`PlotField`"]

g2 = PlotVectorField[{1, 1/10 (10 - x)x}, {t,0,10}, {x,0,10},
  PlotPoints->11, Axes->True, DisplayFunction->Identity];
```

We show both the direction field and one solution:

```
Show[g1,g2];
```



■ Example 4: A Set of Trajectories

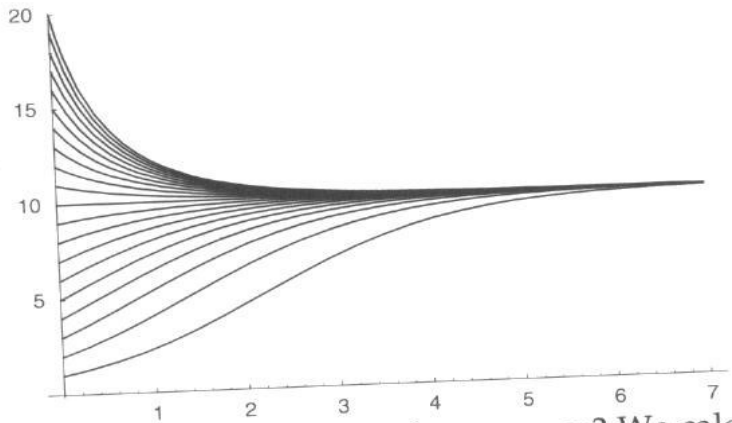
An even clearer plot can be obtained by plotting a set of trajectories, that is, a set of solutions to the equation with different starting points:

```
sol = DSolve[{eq /. {k->1/10, xf->10}, x[0]==x0}, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{10\, E^t\, x0}{10 - x0 + E^t\, x0}\right\}\right\}$$

```
solset = Table[sol, {x0,1,20}];
```

```
g1 = Plot[Evaluate[x[t]/.solset], {t,0,7}, PlotRange->{-0.1,20.1}];
```



What is the inflection point in these curves? We calculate the second derivative and find the value of **t** for which the second derivative is zero:

```
soltt = D[sol,t,t] //Simplify
```

$$\left\{\left\{x''[t] \to \frac{10\,E^t\,(-10+x0)\,x0\,(-10+x0+E^t\,x0)}{(10+(-1+E^t)\,x0)^3}\right\}\right\}$$

```
infl = Solve[(x''[t] /. soltt) == 0, t] //Simplify
```

$$\left\{\{t \to -\infty\},\ \left\{t \to \text{Log}\left[-1+\frac{10}{x0}\right]\right\}\right\}$$

According to the second solution, we get a nonnegative inflection point if **x0** is at most 5. The value of **x** at this point is

```
sol /. infl[[2]] //Simplify
```

$$\left\{\left\{x\left[\text{Log}\left[-1+\frac{10}{x0}\right]\right] \to 5\right\}\right\}$$

that is, for each curve the inflection point is at the fixed level 5.

■ Example 5: Several Solutions

Sometimes the problem has several solutions:

```
eq = {y'[x] == y[x] + x + 1, y[0]^2 - 3y[0] + 2 == 0};
sol = DSolve[eq, y[x], x]
```

$$\{\{y[x] \to -2+3\,E^x-x\},\ \{y[x] \to -2+4\,E^x-x\}\}$$

The two solutions correspond to the two solutions for **y[0]**. Here is a nonlinear equation:

```
DSolve[y'[x] == 1/y[x], y[x], x]
```

$$\left\{\left\{y[x] \to -\sqrt{2}\,\sqrt{x-C[1]}\right\},\ \left\{y[x] \to \sqrt{2}\,\sqrt{x-C[1]}\right\}\right\}$$

■ Example 6: Nonlinear Equations

DSolve can solve many nonlinear equations. Some examples:

```
DSolve[y'[x]^2 == y[x], y[x], x]
```

$$\left\{\left\{y[x] \to \frac{1}{4}\left(x^2-2\,x\,C[1]+C[1]^2\right)\right\}\right\}$$

```
DSolve[y'[x] == y[x]^2 + a, y[x], x]
```

$$\{\{y[x] \to \sqrt{a}\ \text{Tan}[\sqrt{a}\ x + \sqrt{a}\ C[1]]\}\}$$

If we replace the constant **a** in the second example by **x**, then the solution is much more complicated:

```
DSolve[y'[x] == y[x]^2 + x, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{(-1)^{1/3}\,(\text{AiryAiPrime}[(-1)^{1/3}\,x] + \text{AiryBiPrime}[(-1)^{1/3}\,x]\,C[1])}{\text{AiryAi}[(-1)^{1/3}\,x] + \text{AiryBi}[(-1)^{1/3}\,x]\,C[1]}\right\}\right\}$$

■ Example 7: Equations Are Defined by ==

A common problem in solving differential equations is the following:

```
eq = {y'[x] == y[x] + x + 1, y[0] = 1};
sol = DSolve[eq, y[x], x]
```

```
Part::partd : Part specification 1[[1]] is longer than depth of object.
Part::partd : Part specification 1[[2]] is longer than depth of object.
DSolve::nvld :
 The description of the equations appears to be ambiguous or invalid.
DSolve::deqn : Element 1 in the equation list is not an equation.
DSolve[{y'[x] == 1 + x + y[x], 1}, y[x], x]
```

What went wrong? We observe that the initial condition **y[0]=1** is not a correct equation. It must be written as **y[0]==1**. When we wrote **y[0]=1**, we actually assigned the value **1** for **y[0]**, and this causes the error messages. Before we solve the initial value problem, we must clear the value of **y[0]** and correct the initial condition:

```
y[0]=.
eq = {y'[x] == y[x] + x + 1, y[0] == 1};
sol = DSolve[eq, y[x], x]
```

$$\{\{y[x] \to -2 + 3\,E^x - x\}\}$$

22.1.2 Second- and Higher-Order Equations

Here are commands for second-order equations. They generalize directly to higher-order equations.

`DSolve[eq, y[x], x]`	General solution
`sol = DSolve[{eq, y[a]==r, y'[a]==s}, y[x], x]`	Initial value problem
`sol = DSolve[{eq, y[a]==r, y[b]==s}, y[x], x]`	Boundary value problem
`Plot[Evaluate[y[x]/.sol], {x,a,b}]`	Plot the solution

The initial and boundary conditions mentioned here are the simplest ones. The conditions can be more complex.

■ Example 1: Basic Techniques

We ask for a general solution of a second-order equation:

```
eq = y''[x] - y'[x] == 1;
sol = DSolve[eq, y[x], x]
```

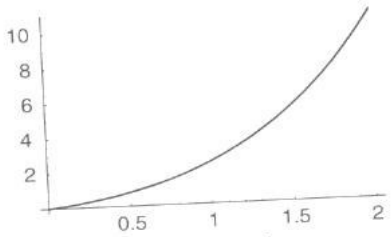
$$\{\{y[x] \to -x + E^x\, C[1] + C[2]\}\}$$

The arbitrary constants are `C[1]` and `C[2]`. Next we give two initial conditions:

```
sol = DSolve[{eq, y[0]==0, y'[0]==1}, y[x], x]
```

$$\{\{y[x] \to -2 + 2\,E^x - x\}\}$$

```
Plot[Evaluate[y[x]/.sol], {x,0,2}];
```

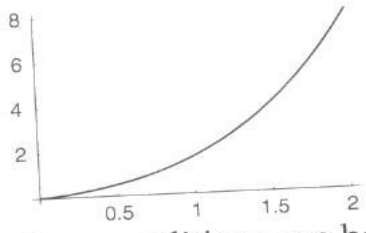


Now we give two boundary conditions, one at $x = 0$ and one at $x = 2$:

```
sol = DSolve[{eq, y[0]==0, y[2]==8}, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{10}{-1+E^2} + \frac{10\,E^x}{-1+E^2} - x\right\}\right\}$$

```
Plot[Evaluate[y[x]/.sol], {x,0,2}];
```

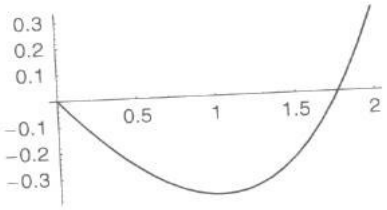


The boundary conditions can be even more complex:

```
sol = DSolve[{eq, y[0] == 0, y[2] + y'[2] == 2}, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{5}{-1+2\,E^2} + \frac{5\,E^x}{-1+2\,E^2} - x\right\}\right\}$$

```
Plot[Evaluate[y[x]/.sol], {x,0,2}];
```



■ Example 2: Constant Coefficients

All linear second-order equations with constant coefficients can be solved. For example,

```
eq = y''[x] == a y'[x] + b y[x] + c;

DSolve[eq, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{c}{b} + E^{\frac{1}{2}\left(a-\sqrt{a^2+4\,b}\right)x}\,C[1] + E^{\frac{1}{2}\left(a+\sqrt{a^2+4\,b}\right)x}\,C[2]\right\}\right\}$$

Note that this is a generic solution. For special values of the parameters the solution can be of another form. The following solution is of the preceding form:

```
DSolve[{eq/.{a->1,b->2,c->1}, y[0]==0, y'[0]==1}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{1}{2}\,E^{-x}\left(-E^{x} + E^{3x}\right)\right\}\right\}$$

but this solution is not:

```
DSolve[{eq/.{a->2,b->-1,c->1}, y[0]==0, y'[0]==1}, y[x], x]
```

$$\{\{y[x] \to 1 - E^{x} + 2\,E^{x}\,x\}\}$$

Next we get powers of –1, but the solution is simplified considerably by using **ComplexExpand**:

```
sol = DSolve[{eq/.{a->1,b->-1,c->1}, y[0] == 0, y'[0] == 1}, y[x], x] //
  Simplify
```

$$\left\{\left\{y[x] \to 1 - E^{(-1)^{1/3}x} - (-1)^{2/3}\,E^{(-1)^{1/3}x} + (-1)^{2/3}\,E^{-(-1)^{2/3}x}\right\}\right\}$$

```
y[x] /. sol //ComplexExpand
```

$$\left\{1 - E^{x/2}\,\text{Cos}\left[\frac{\sqrt{3}\,x}{2}\right] + \sqrt{3}\,E^{x/2}\,\text{Sin}\left[\frac{\sqrt{3}\,x}{2}\right]\right\}$$

■ Example 3: A Set of Trajectories

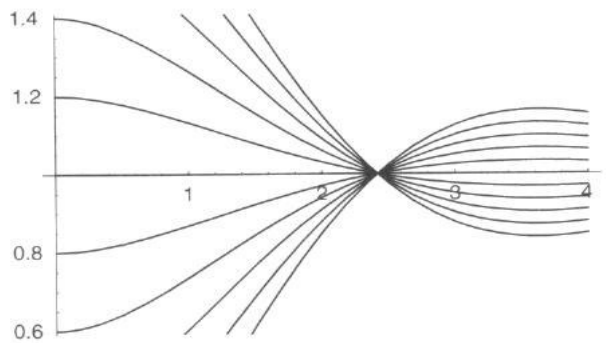
Here is a second-order equation:

```
eq = y''[x] + y'[x] + y[x] == 1;

sol = DSolve[{eq, y[0]==a, y'[0]==b}, y[x], x] //Simplify
```

$$\left\{\left\{y[x] \to E^{-(-1)^{1/3}x}\left(\frac{(-1)^{1/3}\,(-1+a+(-1)^{1/3}\,b)}{1+(-1)^{1/3}} + E^{(-1)^{1/3}x} - \frac{I\,(-(-1)^{1/3} + (-1)^{1/3}\,a + b)\,E^{I\sqrt{3}\,x}}{\sqrt{3}}\right)\right\}\right\}$$

We consider solutions for which **y'[0]** is 0 and **y[0]** takes on values between 0 and 2:

```
solset = Table[sol/.b->0, {a,0,2,0.2}];

Plot[Evaluate[y[x]/.solset], {x,0,4}];
```



■ Example 4: A Third-Order Equation

Here is a third-order equation:

```
eq = y'''[x] - y'[x] == 1;
sol = DSolve[eq, y[x], x]
```

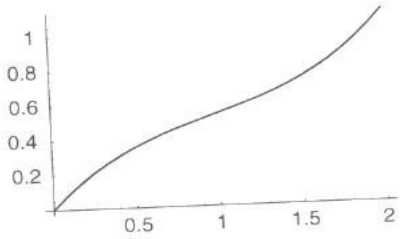
$$\{\{y[x] \to -x - E^{-x} C[1] + E^{x} C[2] + C[3]\}\}$$

We solve an initial value problem:

```
sol = DSolve[{eq, y[0]==0, y'[0]==1, y''[0]==-3/2}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{3}{2} - \frac{7\,E^{-x}}{4} + \frac{E^{x}}{4} - x\right\}\right\}$$

```
Plot[Evaluate[y[x]/.sol], {x,0,2}];
```



■ Example 5: An Implicit Solution

Here is a second-order equation for which the solution is not explicit:

```
eq = y''[x] == 1/y[x]^2;
(sol = DSolve[{eq, y[0]==1, y'[0]==0}, y[x], x]) //Timing
```

```
DSolve::cond :
 Conditioned integral encountered, so some solutions may be not found.
Solve::tdep :
 The equations appear to involve transcendental functions of
   the variables in an essentially non-algebraic way.
```

$$\left\{67.6167\ \text{Second},\ \left\{\text{Solve}\left[-\frac{2\,\text{Log}\left[1+\sqrt{1-\frac{1}{y[x]}}\,\right]+\text{Log}[y[x]]+2\sqrt{1-\frac{1}{y[x]}}\,y[x]}{2\sqrt{2}} == x,\ y[x]\right],\right.\right.$$

$$\left.\left.\text{Solve}\left[\frac{2\,\text{Log}\left[1+\sqrt{1-\frac{1}{y[x]}}\,\right]+\text{Log}[y[x]]+2\sqrt{1-\frac{1}{y[x]}}\,y[x]}{2\sqrt{2}} == x,\ y[x]\right]\right\}\right\}$$

The solution is implicit: if the equations inside the two **Solve** commands could be solved for **y[x]**, we would get two solutions for the differential equation.

Is the solution correct? Let us see. We extract the first equation which implicitly defines the first solution:

```
sol1 = sol[[1,1]]
```

$$-\frac{2\,\text{Log}\left[1+\sqrt{1-\frac{1}{y[x]}}\,\right]+\text{Log}[y[x]]+2\sqrt{1-\frac{1}{y[x]}}\,y[x]}{2\sqrt{2}} == x$$

We take the first and second derivatives:

```
{dsol1 = D[sol1,x]//Simplify,  ddsol1 = D[dsol1,x]//Simplify}
```

$$\left\{-\frac{y'[x]}{\sqrt{2-\frac{2}{y[x]}}} == 1,\ \frac{y'[x]^2}{(2-\frac{2}{y[x]})^{3/2}\,y[x]^2} - \frac{y''[x]}{\sqrt{2-\frac{2}{y[x]}}} == 0\right\}$$

We can now solve **y'[x]** and **y''[x]** from these equations:

```
{dy1 = Solve[dsol1, y'[x]][[1,1]],  ddy1 = Solve[ddsol1/.dy1,
y''[x]][[1,1]]}
```

$$\left\{y'[x] \to -\sqrt{2-\frac{2}{y[x]}},\ y''[x] \to \frac{1}{y[x]^2}\right\}$$

Lastly we substitute the derivatives into the equation and hope for the best:

```
eq /. {dy1,ddy1}                    True
```

The first implicit solution seems to satisfy the equation. We check lastly that the initial conditions are satisfied. We substitute the value 0 for **x** and the value 1 for **y[0]** in **sol1**. The result should be a true statement:

```
sol1 /. x->0 /. y[0]->1             True
```

Then we substitute the same values in **dy1**. The value of **y'[0]** should be 0:

```
dy1 /. x->0 /. y[0]->1              y'[0] -> 0
```

Similarly we could show that the second implicit solution is correct.

22.1.3 Simultaneous Equations

In simultaneous equations we have several dependent variables. Here are some typical commands for two equations. It is a relatively simple procedure to generalize the commands to more equations and different initial and boundary conditions.

DSolve[{eq1,eq2}, {y[x],z[x]}, x]	General solution
sol = DSolve[{eq1,eq2, y[a]==r, z[a]==s}, {y[x],z[x]}, x]	Initial value problem
sol = DSolve[{eq1,eq2, y[a]==r, z[b]==s}, {y[x],z[x]}, x]	Boundary value problem
Plot[Evaluate[{y[x],z[x]}/.sol], {x,a,b}]	Plot **y[x]** and **z[x]**
ParametricPlot[Evaluate[{y[x],z[x]}/.sol], {x,a,b}]	Plot a phase trajectory

■ Example 1: Basic Techniques

First we ask for a general solution:

```
eq = {y'[x] == y[x], z'[x] == 2y[x] + z[x]};

DSolve[eq, {y[x],z[x]}, x]
```

$$\{\{y[x] \to E^x C[1],\ z[x] \to 2E^x x C[1] + E^x C[2]\}\}$$

The solution of this constant coefficient system can also be obtained by the matrix exponential:

```
MatrixExp[{{1,0},{2,1}} x] . {c1,c2}
```

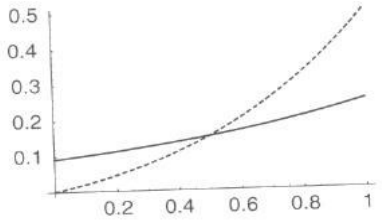
$$\{c1\, E^{x},\ c2\, E^{x} + 2\, c1\, E^{x}\, x\}$$

Here is a boundary value problem:

```
sol = DSolve[Join[eq, {z[0]==0, z'[1]==1}], {y[x],z[x]}, x]
```

$$\left\{\left\{y[x] \to \frac{E^{-1+x}}{4},\ z[x] \to \frac{1}{2} E^{-1+x}\, x\right\}\right\}$$

```
Plot[Evaluate[{y[x],z[x]}/.sol], {x,0,1},
  PlotStyle->{{}, Dashing[{0.012}]}];
```



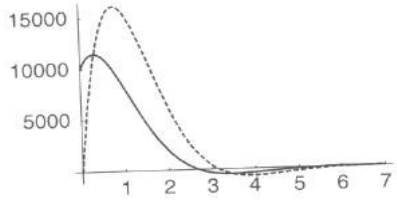
■ Example 2: Phase Trajectories

Consider the following system:

```
eq = {y'[x] == y[x] - z[x], z'[x] == 5y[x] - 3z[x],
      y[0] == 10000, z[0] == 0};
sol = DSolve[eq, {y[x],z[x]}, x] //FullSimplify
```

$$\{\{y[x] \to 10000\, E^{-x}\, (\text{Cos}[x] + 2\, \text{Sin}[x]),\ z[x] \to 50000\, E^{-x}\, \text{Sin}[x]\}\}$$

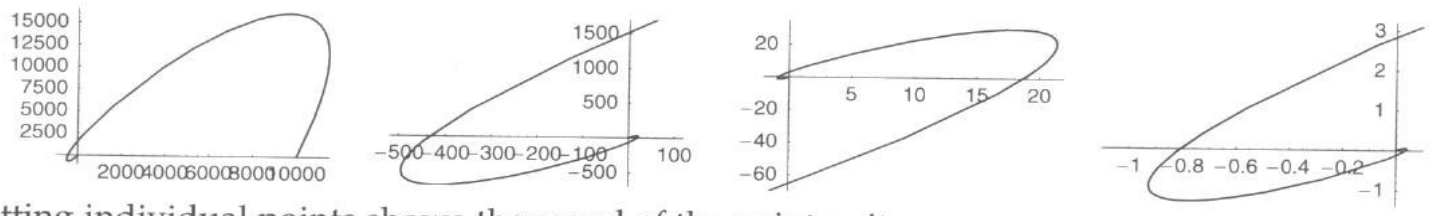
```
Plot[Evaluate[{y[x],z[x]}/.sol], {x,0,7},
  PlotStyle->{{}, Dashing[{0.012}]},
  Ticks->{Range[7], {5000,10000,15000}}];
```



With **ParametricPlot** we can plot phase trajectories for equations with two dependent variables. A plot of this kind describes how the point **{y[x],z[x]}** moves on the (y, z) plane. We plot four figures of the solution **sol**. These figures show how the curve approaches the point (0, 0) like a spiral:

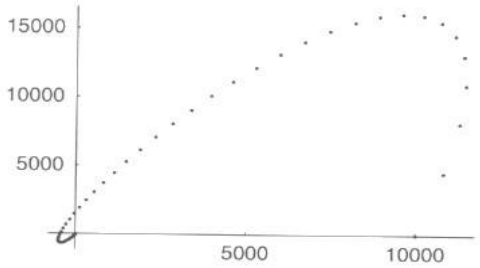
```
Map[ParametricPlot[Evaluate[{y[x],z[x]}/.sol],{x,0,40},
  PlotRange->#, DisplayFunction->Identity]&,
  { {{-1000,11500},{-1000,16500}}, {{-520,120},{-700,1700}},
    {{-2,22},{-70,35}}, {{-1.1,0.1},{-1.4,3.2}} }];
```

```
Show[GraphicsArray[%]];
```



Plotting individual points shows the speed of the point as it moves on a curve:

```
Table[Evaluate[{y[x],z[x]}/.sol[[1]]],{x,0,6,0.1}];

ListPlot[%, Ticks->{{5000, 10000},{5000,10000,15000}}];
```



■ Example 3: A Set of Trajectories

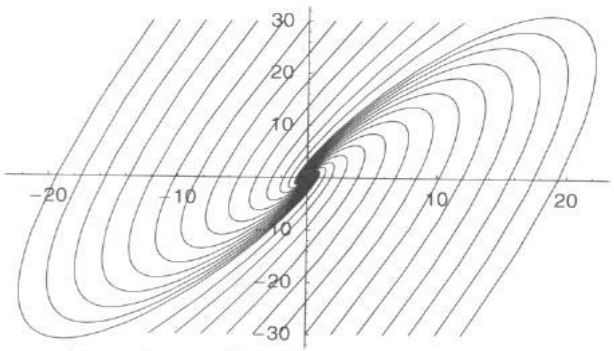
Plotting several trajectories from different starting points gives a good description of the behavior of the system. We consider the same system again:

```
eq = {y'[x] == y[x] - z[x], z'[x] == 5y[x] - 3z[x]};

sol = DSolve[Join[eq, {y[0]==y0, z[0]==z0}], {y[x],z[x]}, x]//
  FullSimplify
```

```
{{y[x] → E^-x (y0 Cos[x] + (2 y0 - z0) Sin[x]),
  z[x] → E^-x (z0 Cos[x] + (5 y0 - 2 z0) Sin[x])}}
```

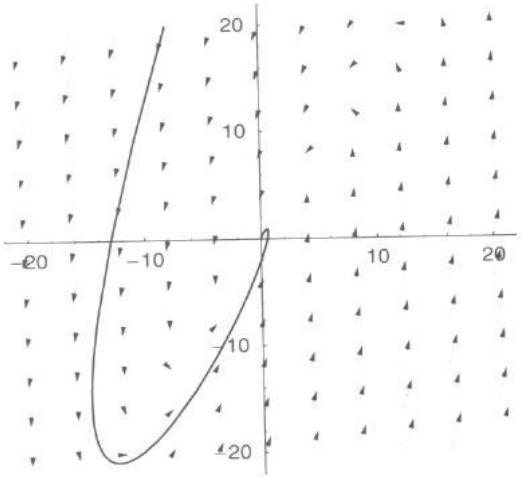
We let **y0** vary from –12 to 12 in steps of 2, and then we give **z0** first the value 30 and then the value –30:

```
solset = Table[sol[[1]], {y0,-12,12,2}];

Map[ParametricPlot[Evaluate[{y[x],z[x]}/.solset/.z0->#], {x,0,6},
  PlotStyle->AbsoluteThickness[0.25], DisplayFunction->Identity]&,
  {30,-30}];

Show[%, DisplayFunction->$DisplayFunction];
```



We can also plot a direction field:

```
sol = DSolve[Join[eq, {y[0]==-8, z[0]==20}], {y[x],z[x]}, x];

Needs["Graphics`PlotField`"]

Block[{$DisplayFunction = Identity},
  g1 = PlotVectorField[{u - v, 5u - 3v}, {u,-20,20}, {v,-20,20},
    PlotPoints->11];
  g2 = ParametricPlot[Evaluate[{y[x],z[x]}/.sol], {x,0,6}]];

Show[g1,g2, Axes->True];
```



If we have three equations, then **ParametricPlot3D** can be used to plot trajectories in three dimensions; for an example see Section 22.3.3.

22.2 More about Symbolic Solutions

22.2.1 Calculating with the Solution

If we want to calculate with the solution, then the form of the solution considered in Section 22.1 is not suitable, as will be seen shortly. In this section we consider other forms for the solution which can be used in calculations.

■ Solution as a Pure Function

Thus far we have asked for the solution to **y[x]**. The solution is then in a familiar form, but if we want to do calculations with the solution, then this form is not suitable. As an example, solve an equation and ask for its value at some point:

```
eq = {y'[x] == y[x] + x + 1, y[0] == 1};

sol1 = DSolve[eq, y[x], x]
```

$$\{\{y[x] \to -2 + 3\,E^{x} - x\}\}$$

```
y[0] /. sol1[[1]]
```

$$y[0]$$

We did not obtain the value of **y[0]**. In fact, the solution for **y[x]** knows the value of **y** only for **x** and not for any other argument like **0**. There is a special form of **DSolve** that gives the solution in a form suitable for calculations.