

## Part IV. Linear Differential Equations

### Section 2. State Space

The motion of an unforced linear oscillator

$$a x'' + b x' + c x = 0 \quad (a, b, c \text{ constants})$$

is completely determined by its initial position and velocity; time does not matter because the system is autonomous. The rate of change of position,

$$x' = v$$

and velocity,

$$v' = (-c x - b v)/a$$

only depend upon position and velocity. Because of this, phase plane trajectories for the unforced oscillator cannot overlap. See Simmons/Krantz, Chapter 11, Section 2.

In this section we continue to analyze second order linear equations, autonomous and forced. Exact solutions will be obtained, followed by numeric solutions using **DEplot** and **dsolve**. The phase plane plays an important role in the analysis, and state space is introduced to help understand solutions to the forced equation.

#### *Autonomous equations*

Families of solution curves for a second order equation can be obtained by solving the equation in terms of generic initial values, say  $x(0) = x_0$ ,  $x'(0) = v_0$ . The solution formula is made into a function  $\phi$  of the parameters  $x_0$ ,  $v_0$ , and time  $t$ :

$$x = \phi(x_0, v_0, t).$$

This formula gives position at time  $t$ . The partial derivative of  $\phi$  with respect to  $t$  provides the velocity formula.

$$v = \frac{\partial}{\partial t} \phi(x_0, v_0, t)$$

*Example.* Obtain a phi function for the damped mass spring system

$$x'' + 0.2 x' + x = 0$$

Use it to plot time series for  $x$  and  $v$  and trajectories in the phase plane.

Begin by entering the equation and solving the IVP with generic initial values (at  $t = 0$ ).

```
> DE1 := diff(x(t),t,t) + 0.2*diff(x(t),t) + x(t) = 0;  
soln1 := dsolve( {DE1, x(0)=x0, D(x)(0)=v0} );
```

$$DE1 := \frac{d^2}{dt^2} x(t) + 0.2 \left[ \frac{d}{dt} x(t) \right] + x(t) = 0$$

$$soln1 := x(t) = \frac{1}{33} (10 v_0 + x_0) \sqrt{11} e^{-\frac{1}{10} t} \sin\left(\frac{3}{10} \sqrt{11} t\right) + x_0 e^{-\frac{1}{10} t} \cos\left(\frac{3}{10} \sqrt{11} t\right) \quad (1)$$

The **unapply** procedure makes the solution formula into the function phi.

```
> phi := unapply(rhs(soln1), x0, v0, t);
```

$$\phi := (x0, v0, t) \rightarrow \frac{1}{33} (10 v0 + x0) \sqrt{11} e^{-\frac{1}{10} t} \sin\left(\frac{3}{10} \sqrt{11} t\right) + x0 e^{-\frac{1}{10} t} \cos\left(\frac{3}{10} \sqrt{11} t\right) \quad (2)$$

The system is underdamped with time constant 10 seconds. Therefore, in about 40 seconds, the plot of any solution will disappear from view.

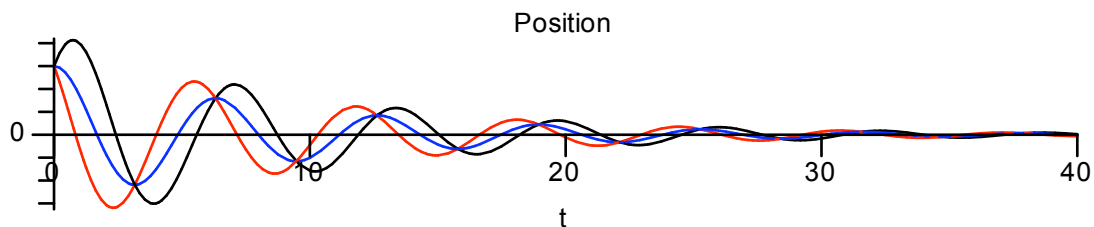
The first plot is a family of three time series for position, using the initial conditions

$$x(0) = 3, v(0) = -3, 0, 3.$$

```
> plots[setoptions]( tickmarks = [5,5] );
```

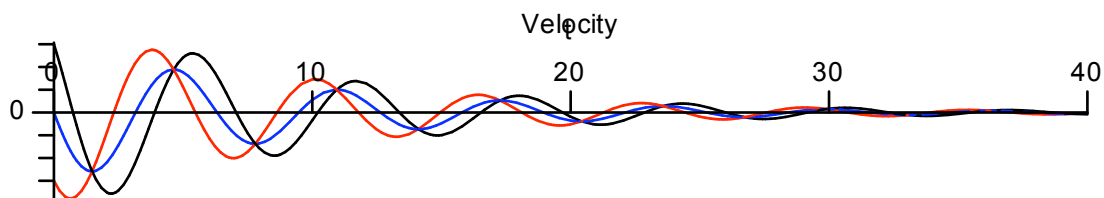
Note that **plot** fails to put any values on the vertical axis. This is a bug in Maple 10.

```
> plot( [ phi(3, 3*k, t)$k=-1..1 ], t=0..40, color=[red,blue,black],
        title="Position");
```



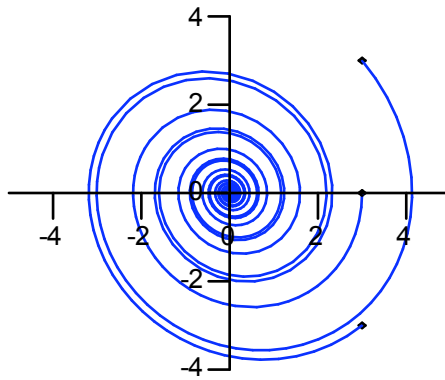
Here are the corresponding velocity curves.

```
> plot( [ diff(phi(3, 3*k, t), t)$k=-1..1 ], t=0..40, color=[red,blue,black]
        title="Velocity");
```



The phase plane trajectories are drawn below in green. The three initial points have also been plotted.

```
> inits := [ [3, 3*k]$k=-1..1 ]:
plot( [ [phi(3, 3*k, t), diff(phi(3, 3*k, t), t), t=0..40]$k=-1..1,
        inits ], style=[line$3, point$3], color=[blue$3, black$3],
        view=[-5..5, -4..4], scaling=constrained);
```



The trajectories get close, but they do not intersect.

### ***Force it: A non-autonomous equation***

Now force the system periodically with the cosine function.

$$x'' + 0.2x' + x = \cos(t)$$

The equation is no longer autonomous and, as we shall see below, the phase plane trajectories overlap.

```
> DE2 := diff(x(t),t,t) + 0.2*diff(x(t),t) + x(t) = cos(t);
soln2 := dsolve( {DE2, x(0)=x0, D(x)(0)=v0} );
```

$$DE2 := \frac{d^2}{dt^2} x(t) + 0.2 \left[ \frac{d}{dt} x(t) \right] + x(t) = \cos(t)$$

$$soln2 := x(t) = \frac{1}{33} e^{-\frac{1}{10}t} \sin\left(\frac{3}{10}\sqrt{11}t\right) (10v_0 - 50 + x_0)\sqrt{11} + x_0 e^{-\frac{1}{10}t} \cos\left(\frac{3}{10}\sqrt{11}t\right) + 5 \sin(t) \quad (3)$$

The new phi function is defined, output suppressed.

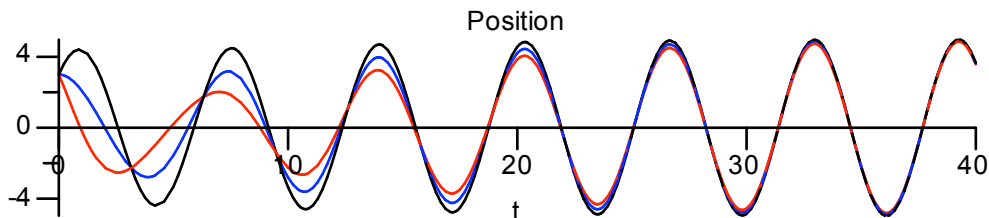
```
> phi := unapply(rhs(soln2), x0, v0, t):
```

The time series for  $x$  with the same initial conditions:

$$x(0) = 3, v(0) = -3, 0, 3.$$

are drawn first.

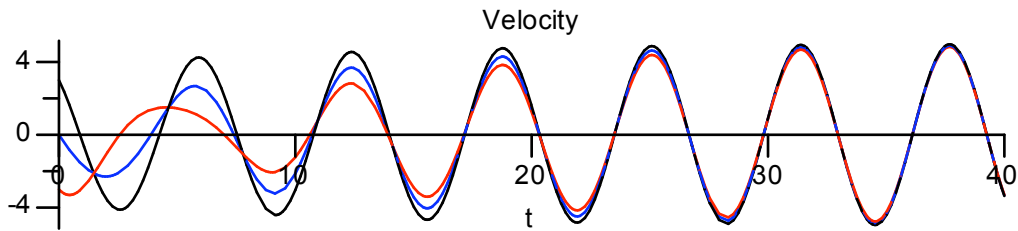
```
> plot( [ phi(3,3*k,t)$k=-1..1], t=0..40, color=[red,blue,black],
title="Position");
```



The velocity curves look like this.

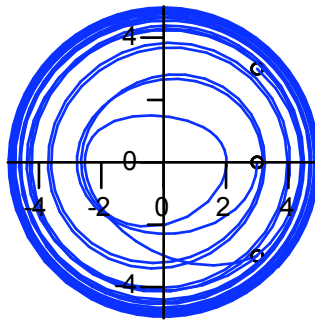
```
> plot( [ diff(phi(3,3*k,t),t)$k=-1..1], t=0..40, color=[red,blue,black]
```

```
title="Velocity");
```



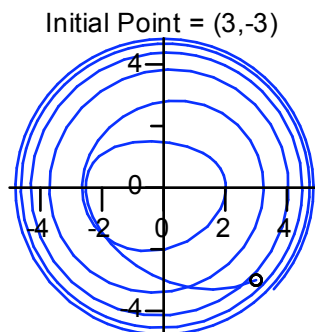
The phase plane trajectories follow and, as you can see, they are all tangled up. Note that the symbol marking the initial points has been made into a large circle using `symbol=circle` and `symbolsize=18`.

```
> plot( [ [phi(3,3*k,t),diff(phi(3,3*k,t),t),t=0..40]$k=-1..1,
          inits ], style=[line$3,point$3], color=[blue$3,black$3],
          view=[-5..5,-5..5], scaling=constrained, symbol=circle,
          symbolsize=18);
```



Even one phase plane trajectory gets tangled as it winds around overlapping itself. The trajectory for  $x(0) = 3$ ,  $v(0) = -3$  is shown below.

```
> plot( [ [phi(3,-3,t),diff(phi(3,-3,t),t),t=0..40],
          [[3,-3]] ], style=[line,point], color=[blue,black],
          view=[-5..5,-5..5], scaling=constrained, symbol=circle,
          symbolsize=18, title="Initial Point = (3,-3)");
```



Overlapping phase plane trajectories reflect the fact that solutions are no longer uniquely determined by position and velocity. Time must also be taken into account to determine the future of the system. Geometrically, this is accomplished by pulling the trajectory to 3 dimensional  $x,v,t$  space where time is used as the third coordinate. This is called *state space* and the parametrized curve is called a *state space trajectory*.

### ***State space trajectories: The spacecurve procedure***

The state space trajectory for a second order differential equation is the parametrized curve

$$x = x(t), \quad v = x'(t), \quad t = t$$

plotted in  $x, v, t$ -space. Three dimensional trajectories are plotted in Maple using the **spacecurve** procedure (plots package). The syntax for a 3 dimensional curve is

```
spacecurve( [f(t),g(t),h(t),t=a..b], options )
```

The plots package is loaded first.

```
> with(plots):
```

The phase plane trajectory in the last plot corresponds to the initial state

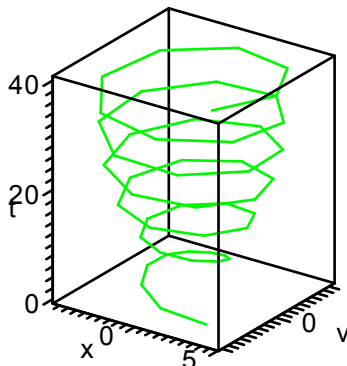
$$x0 = 3, \quad v0 = -3, \quad t0 = 0$$

The state space trajectory is plotted below. First, however, to save repetitious typing, we define a sequence of optional equations that will be used for the spacecurves in this section.

```
> SOptions := color=green, axes=boxed, labels=["x","v","t"],  
orientation=[-55,70]:
```

The equation `orientation = [-55, 70]` is to give a nice viewing angle:  $-55$  degrees from the positive  $x$ -axis and  $70$  degrees from the positive vertical axes. All state space trajectories will be shown using these settings.

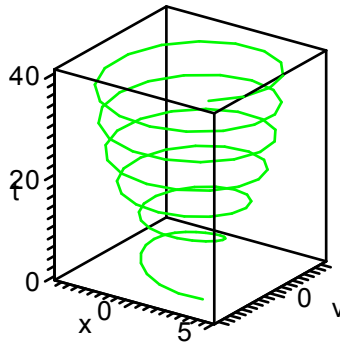
```
> spacecurve( [phi(3,-3,t),diff(phi(3,-3,t),t),t,t=0..40],SOptions);
```



- As the default, **spacecurve** plots 50 points. This is not enough for a smooth curve in this example. This can be fixed by adding an optional equation of the form `numpoints = n`.

In the following plot we have entered `numpoints = 100`.

```
> spacecurve( [phi(3,-3,t),diff(phi(3,-3,t),t),t,t=0..40], SOptions,  
numpoints=100);
```



### Numerical solutions: DEplot

The **DEplot** procedure can be used to create phase plane pictures displaying one or many trajectories. If the equation is autonomous, tangent vectors can also be displayed. The display of several trajectories is called a "flow". In order to make trajectories, **DEplot** must be supplied with the equivalent system of two first order equations (recall the example at the end of Section 1).

The following entry creates a procedure named *DEsystem* to convert a second order differential equation into an equivalent system of first order equations. The input consists of the second order equation, the dependent variable, the derivative variable, and the independent variable. The output is a set containing the two first order equations.

```
> DEsystem := proc(DE, x, v, t)
  local de1, de2;
  de1 := diff(x(t), t) = v(t);
  subs(de1, DE);
  de2 := isolate(%, diff(v(t), t));
  {de1, de2};
end proc;
```

```
DEsystem := proc(DE, x, v, t) (4)
```

```
  local de1, de2;
  de1 := diff(x(t), t) = v(t);
  subs(de1, DE);
  de2 := isolate(`%`, diff(v(t), t));
  {de1, de2}
```

**end proc**

First test *DEsystem* on the equation named *DE1*.

```
> This_is_DE1, DE1;
DEsystem(DE1, x, v, t);
```

$$\textit{This\_is\_DE1}, \frac{d^2}{dt^2} x(t) + 0.2 \left( \frac{d}{dt} x(t) \right) + x(t) = 0$$

$$\left\{ \frac{d}{dt} v(t) = -0.2 v(t) - x(t), \frac{d}{dt} x(t) = v(t) \right\} \quad (5)$$

And then on the equation *DE2*.

```
> This_is_DE2,DE2;
DEsystem(DE2,x,v,t);
```

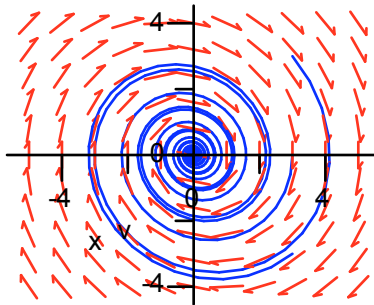
$$\begin{aligned} \text{This\_is\_DE2, } \frac{d^2}{dt^2} x(t) + 0.2 \left( \frac{d}{dt} x(t) \right) + x(t) &= \cos(t) \\ \left\{ \frac{d}{dt} v(t) = \cos(t) - 0.2 v(t) - x(t), \frac{d}{dt} x(t) = v(t) \right\} \end{aligned} \quad (6)$$

As soon as the **DEtools** package is loaded we can plot some trajectories.

```
> with(DEtools):
```

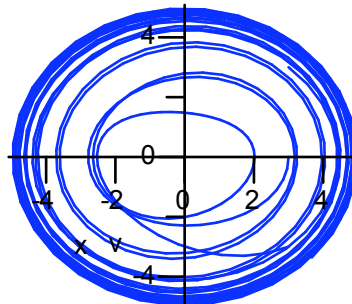
The first example is the first flow created earlier using the exact solution obtained by **dsolve**. **DEplot** draws arrows because *DE1* is autonomous.

```
> DEplot( DEsystem( DE1,x,v,t),[x(t),v(t)],t=0..40,x=-5..5,v=-4..4,
{[x(0)=3,v(0)=3*k]$k=-1..1},stepsize=0.2,dirgrid=[11,11],
linecolor=blue, thickness=1);
```



Here is **DEplot's** rendition of the tangled trajectories associated with the non autonomous equation *DE2*. **DEplot** cannot draw phase plane arrows because the system is not autonomous. There are an infinite number of arrows for each point in phase space.

```
> DEplot( DEsystem( DE2,x,v,t),[x(t),v(t)],t=0..40,x=-5..5,v=-5..5,
{[x(0)=3,v(0)=3*k]$k=-1..1},stepsize=0.2,dirgrid=[11,11],
linecolor=blue, thickness=1);
```

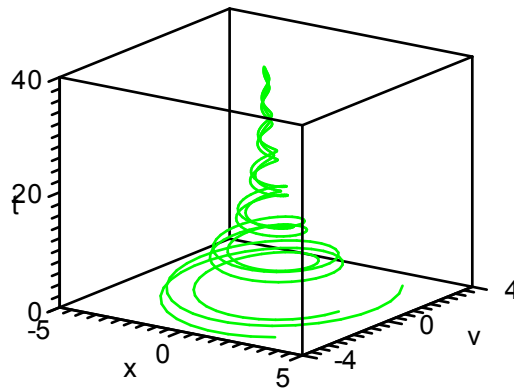


The procedure named **DEplot3d** will draw 3 dimensional state space trajectories. Trajectories for *DE1* are shown below. The optional equation

```
scene=[y,v,t]
```

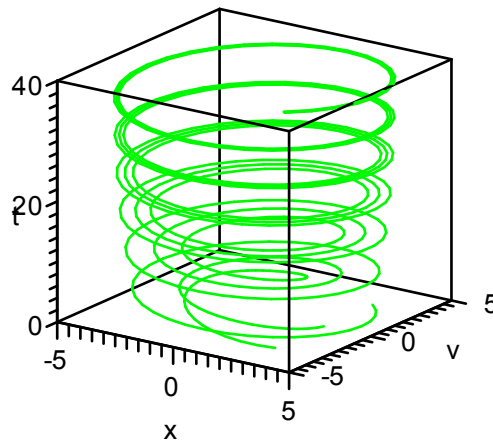
is required to force **DEplot3d** to plot time as the third variable (the default is  $[t,y,v]$ ).

```
> DEplot3d(DEsystem( DE1,x,v,t),[x(t),v(t)],t=0..40,x=-5..5,v=-4..4,
  {[x(0)=3,v(0)=3*k]$k=-1..1}, stepsize=0.2,
  linecolor=green, scene=[x,v,t], orientation=[-55,70],
  thickness=1);
```



The state space trajectories for *DE2* look like this.

```
> DEplot3d( DEsystem(DE2,x,v,t),[x(t),v(t)],t=0..40,x=-5..5,v=-5..5,
  {[x(0)=3,v(0)=3*k]$k=-1..1}, stepsize=0.2,
  linecolor=green, scene=[x,v,t], orientation=[-55,70],
  thickness=1);
```



### **Numerical solutions: dsolve**

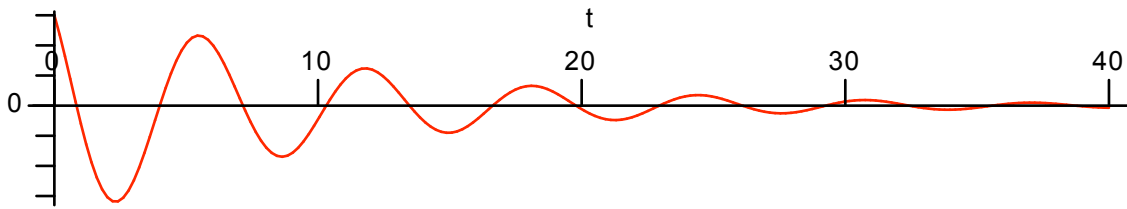
The **dsolve/numeric** procedure can also make pictures like these. Use it in combination with **odeplot** (plots package). It only handles one IVP at a time, and it does not make arrows. The following examples apply **dsolve/numeric** to IVPs with  $x(0) = 3, x'(0) = -3$ .

Applied to *DE1*:

```
> num_soln1 := dsolve( {DE1,x(0)=3,D(x)(0)=-3}, x(t), numeric);
  num_soln1 := proc(x_rkf45) ... end proc (7)
```

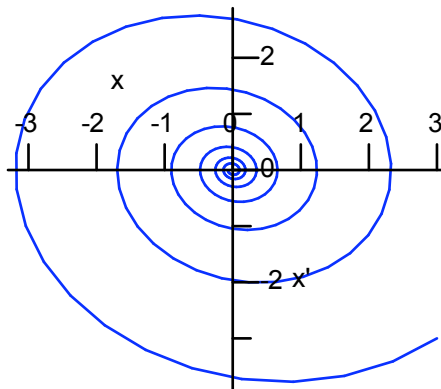
A time series for *x*.

```
> odeplot( num_soln1, [t,x(t)], t=0..40, numpoints=200 );
```



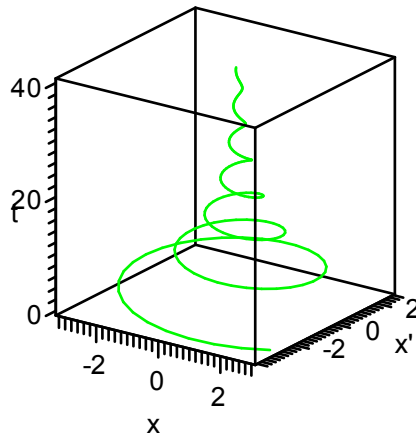
The phase plane trajectory.

```
> odeplot( num_soln1, [x(t),diff(x(t),t)], t=0..40, numpoints=200,
           color=blue);
```



The state space trajectory.

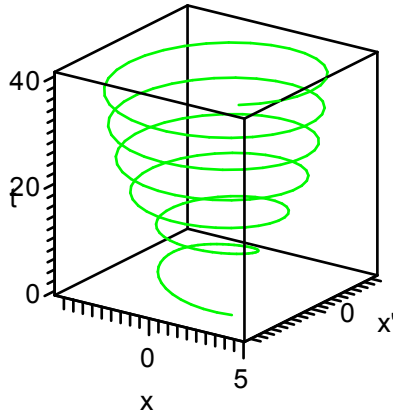
```
> odeplot( num_soln1, [x(t),diff(x(t),t),t], t=0..40, numpoints=200,
           color=green, axes=boxed, orientation=[-55,70]);
```



Now applied to *DE2*, the state space trajectory.

```
> num_soln2 := dsolve( {DE2,x(0)=3,D(x)(0)=-3}, x(t), numeric);
odeplot( num_soln2, [x(t),diff(x(t),t),t], t=0..40, numpoints=200,
         color=green, axes=boxed, orientation=[-55,70]);

num_soln2 := proc(x_rkf45) ... end proc
```



The principal advantage to using **dsolve** is the fact that it can produce numerical output. You need numbers, we got numbers. Here, for example, are the approximate values of  $x$  and  $x'$  for the *DE2* initial value problem when  $t = 2$ .

> **num\_soln2(2);**

$$\left[ t=2., x(t) = -2.24064830891912648, \frac{d}{dt} x(t) = -1.06843334242101906 \right] \quad (8)$$

An array of approximate solution values to this problem is obtained using the equation

$$\text{output}=\text{array}([ t1, t2, \dots , tn ]).$$

Note that the output is suppressed, then displayed in compact 4 digit form.

> **dsolve( {DE2,x(0)=3,D(x)(0)=-3}, x(t), numeric,  
output=array([k/2\$ k=0..6]) ):  
evalf[4](%);**

$$\left[ \begin{array}{ccc} t & x(t) & \frac{d}{dt} x(t) \\ 0. & 3. & -3. \\ 0.5000 & 1.392 & -3.303 \\ 1. & -.1878 & -2.918 \\ 1.500 & -1.451 & -2.082 \\ 2. & -2.241 & -1.068 \\ 2.500 & -2.529 & -.1092 \\ 3. & -2.384 & 0.6485 \end{array} \right] \quad (9)$$