Part IV. Linear Differential Equations

Section 2. State Space

The motion of an unforced linear oscillator

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

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

$$y' = v$$

and velocity,

$$v' = (-c\,y - b\,v)/a$$

only depend upon position and velocity. Because of this, phase plane trajectories for the unforced oscillator cannot overlap. See Ledder, Chapter 5, Theorem 5.3.1.

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 $y(0) = y0$, $y'(0) = v0$. The solution formula is made into a function phi of the parameters $y0$, $v0$, and time t:

$$y = \phi(y0, v0, t)\ .$$

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

$$v = \frac{\partial}{\partial t}\,\phi(y0, v0, t)$$

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

$$y'' + 0.2\,y' + y = 0$$

Use it to plot time series for y 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(y(t),t,t) + 0.2*diff(y(t),t) + y(t) = 0;
  soln1 := dsolve( {DE1, y(0)=y0, D(y)(0)=v0} );
```

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

$$soln1 := y(t) = \frac{1}{33}\sqrt{11}\,(10\,v0 + y0)\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\sin\left(\frac{3}{10}\sqrt{11}\,t\right) + y0\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\cos\left(\frac{3}{10}\sqrt{11}\,t\right)$$

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

```
> phi := unapply(rhs(soln1),y0,v0,t);
```

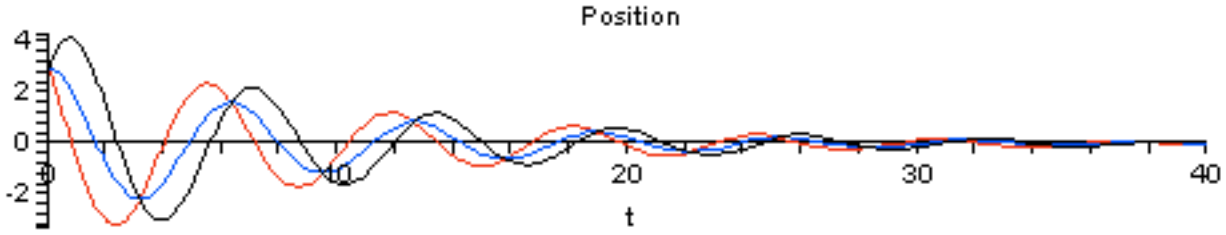
$$\phi := (y0, v0, t) \rightarrow \frac{1}{33}\sqrt{11}\,(10\,v0 + y0)\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\sin\left(\frac{3}{10}\sqrt{11}\,t\right) + y0\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\cos\left(\frac{3}{10}\sqrt{11}\,t\right)$$

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

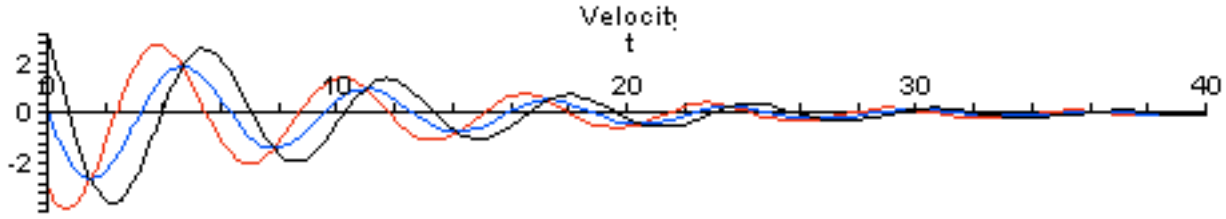
$$y(0) = 3,\ v(0) = -3,\ 0,\ 3.$$

```
> plot( [ phi(3,3*k,t)$k=-1..1], t=0..40, color=[red,blue,black],
  ytickmarks=4,
          title="Position");
```



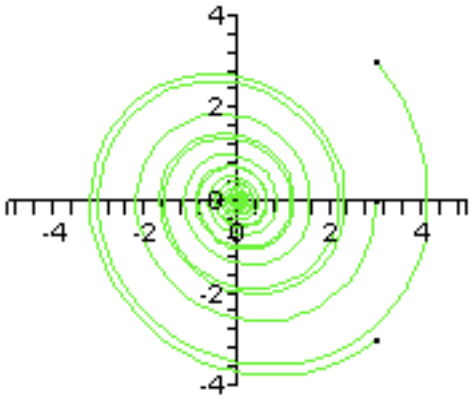
Here are the corresponding velocity curves.

```
> plot( [ diff(phi(3,3*k,t),t)$k=-1..1], t=0..40, color=[red,blue,black],
          ytickmarks=4, title="Velocity");
```



The phase plane trajectories are drawn below. 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=[green$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.

$$y'' + 0.2\,y' + y = \cos(t)$$

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

```
> DE2 := diff(y(t),t,t) + 0.2*diff(y(t),t) + y(t) = cos(t);
  soln2 := dsolve( {DE2, y(0)=y0, D(y)(0)=v0} );
```

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

$$soln2 := y(t) = \frac{1}{33}\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\sin\left(\frac{3}{10}\sqrt{11}\,t\right)\sqrt{11}\,(10\,v0 - 50 + y0) + y0\,\mathbf{e}^{\left(-\frac{1}{10}t\right)}\cos\left(\frac{3}{10}\sqrt{11}\,t\right) + 5\sin(t)$$

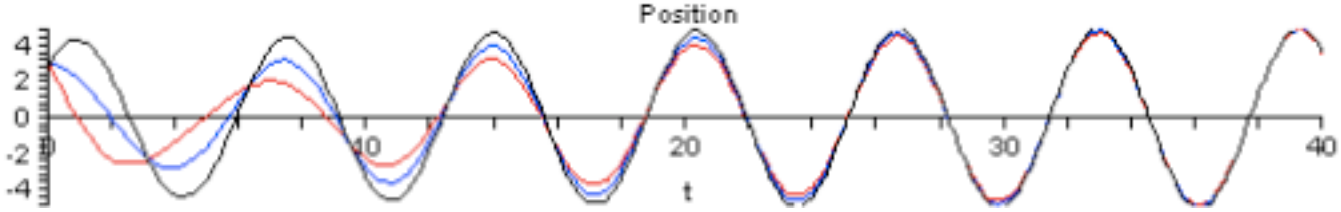
The new phi function, output suppressed.

```
> phi := unapply(rhs(soln2),y0,v0,t):
```

The time series for y with the same initial conditions:

$$y(0) = 3,\ v(0) = -3,\ 0,\ 3.$$

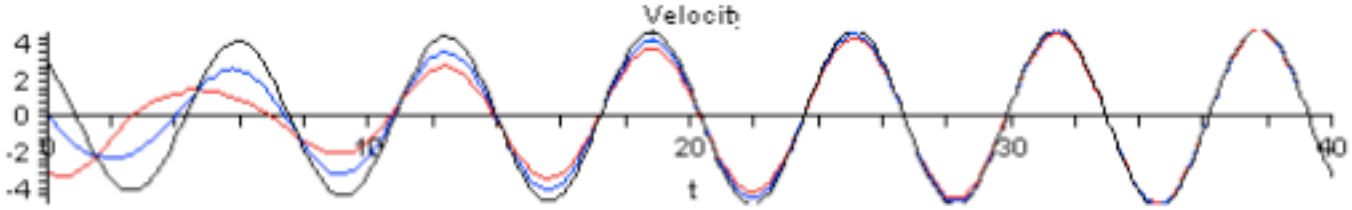
```
> plot( [ phi(3,3*k,t)$k=-1..1], t=0..40, color=[red,blue,black],
        title="Position");
```



The velocity curves:

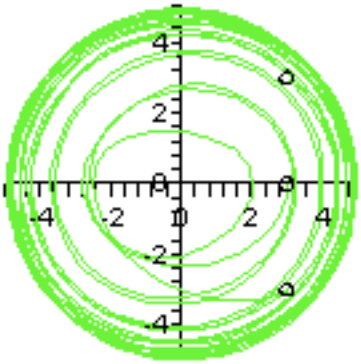
```
> plot( [ diff(phi(3,3*k,t),t)$k=-1..1], t=0..40, color=[red,blue,black],
```

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



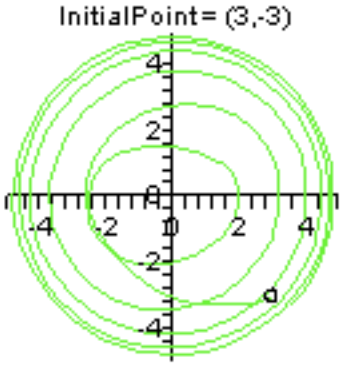
And the phase plane trajectories, all tangled up. Note that the symbol marking the initial points has been made into a large circle using **symbol=circle** and **symbolsize=18** .

```
> 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=[green$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 y(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=[green,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 y,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

$$y = y(t),\ v = y'(t),\ t = t$$

plotted in y,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 )
```

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

```
Warning, the name changecoords has been redefined
```

The phase plane trajectory above corresponds to the initial state

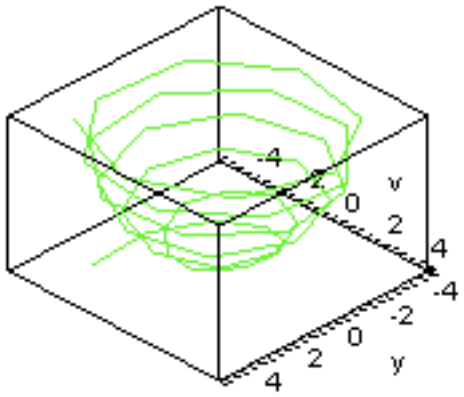
$$y0 = 3,\ v0 = -3,\ t0 = 0$$

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

```
> SCset := color=green, axes=boxed, labels=["y","v","t"]:
```

The promised state space trajectory is shown using these settings.

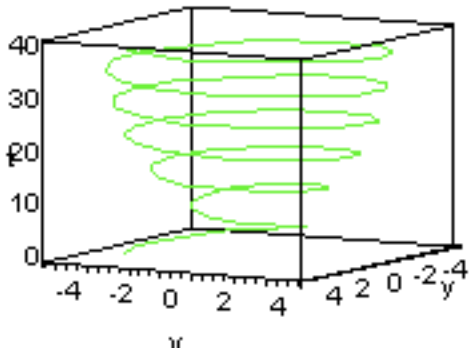
```
> spacecurve( [phi(3,-3,t),diff(phi(3,-3,t),t),t,t=0..40], SCset);
```



- As the default, **spacecurve** plots 50 points and displays the curve from the viewpoint of the spherical coordinate angles theta = 45, phi = 45 (degrees). (Recall that in spherical coordinates, theta is the angle from the positive x axis, and phi is the angle from the positive z axis.)

The next entry adds the equation **numpoints = 100** to get a smoother curve and the equation **orientation=[30,80]** to change the viewpoint to the spherical angles theta = 30, phi = 80. This view shows that state space trajectory does not overlap.

```
> spacecurve( [phi(3,-3,t),diff(phi(3,-3,t),t),t,t=0..40], SCset,
              numpoints=100, orientation=[30,80]);
```



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 two first order equations. The input consists of the second order equation, the dependent variable, its derivative variable, and the independent variable. The output is a set containing the two first order equations.

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

```
DEsystem := proc(DE, y, v, t)
local de1, de2;
      de1 := diff(y(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,y,v,t);
```

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

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

And then on DE2.

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

$$This\_is\_DE2, \left(\frac{d^2}{dt^2} y(t) \right) + 0.2 \left(\frac{d}{dt} y(t) \right) + y(t) = \cos(t)$$

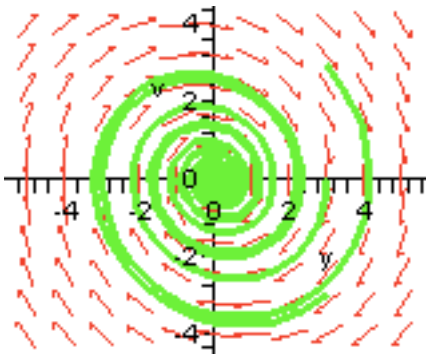
$$\left\{ \frac{d}{dt} v(t) = \cos(t) - 0.2\, v(t) - y(t), \frac{d}{dt} y(t) = v(t) \right\}$$

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

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

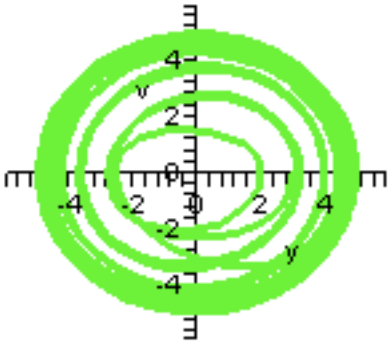
The first example is the first flow created earlier using the exact solution obtained by **dsolve**. **DEplot** draws arrows because DE1 is autonomous.

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



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 phase point.

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



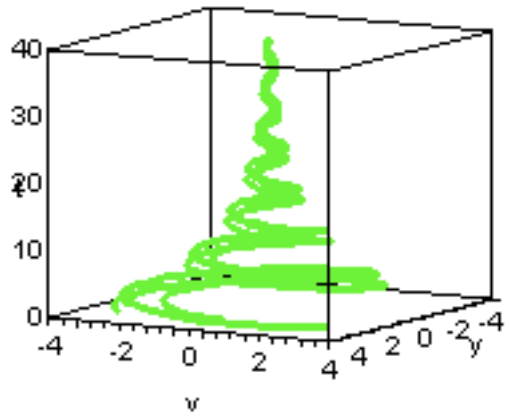
The procedure named **DEplot3d** will draw 3 dimensional state space trajectories. Trajectories for DE1 are

drawn below. The optional equation

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

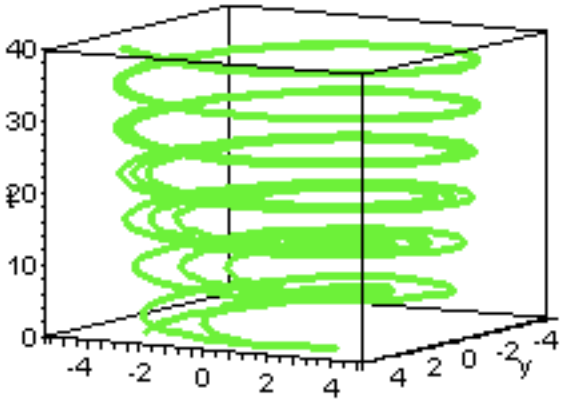
is required to force **DEplot3d** to plot time as the third variable (the default is [*t*,*y*,*v*]).

```
> DEplot3d( DEsystem(DE1,y,v,t), [y(t),v(t)], t=0..40, y=-5..5, v=-4..4,
          {[y(0)=3,v(0)=3*k]$k=-1..1}, stepsize=0.2,
           linecolor=green, scene=[y,v,t], orientation=[30,80]);
```



The state space trajectories for DE2 look like this.

```
> DEplot3d( DEsystem(DE2,y,v,t), [y(t),v(t)], t=0..40, y=-5..5, v=-5..5,
          {[y(0)=3,v(0)=3*k]$k=-1..1}, stepsize=0.2,
           linecolor=green, scene=[y,v,t], orientation=[30,80]);
```



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 $y(0) = 3$, $y'(0) = -3$.

Applied to DE1:

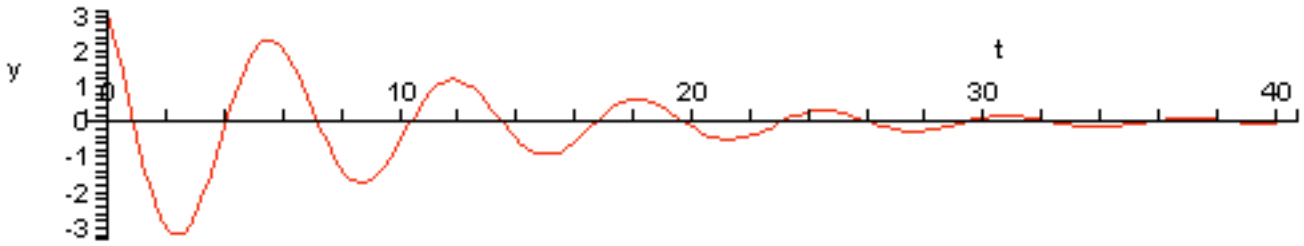
```
> num_soln1 := dsolve( {DE1,y(0)=3,D(y)(0)=-3}, y(t), numeric);
```

num_soln1 := **proc**(*x_rkf45*) ... **end proc**;

A time series for y.

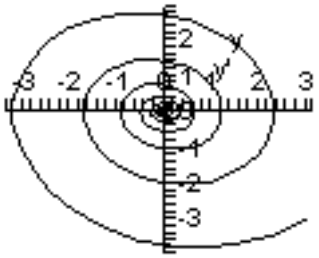
```
> with(plots):
  odeplot( num_soln1, [t,y(t)], t=0..40, numpoints=200, color=red);
```

```
Warning, the name changecoords has been redefined
```



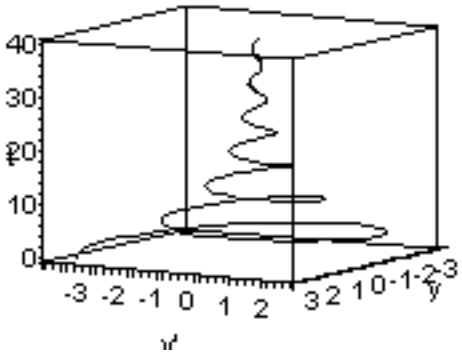
The phase trajectory.

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



The state space trajectory.

```
> odeplot( num_soln1, [y(t),diff(y(t),t),t], t=0..40, numpoints=200,
           color=black, axes=boxed, orientation=[30,80]);
```

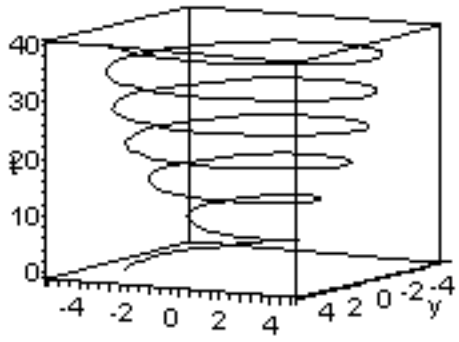


Applied to DE2, the state space trajectory.

```
> num_soln2 := dsolve( {DE2,y(0)=3,D(y)(0)=-3}, y(t), numeric);
  odeplot( num_soln2, [y(t),diff(y(t),t),t], t=0..40, numpoints=200,
```

```
color=black, axes=boxed, orientation=[30,80]);
```

$num\_soln2 := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc}$;



The big advantage to using **dsolve** is the fact that it can produce numerical output. You need numbers, we got numbers.

Here are the approximate values of y and y' for the DE2 initial value problem when $t = 2$.

```
> num_soln2(2);
```

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

An array of approximate solution values to the DE2 initial value problem is obtained using **output=array([...])**. Note that the output is suppressed, then displayed in compact 4 digit form.

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

$$\left[\left[t, y(t), \frac{d}{dt}y(t) \right],\ \left[\begin{array}{rrr} 0. & 3. & -3. \\ 0.5000 & 1.392 & -3.303 \\ 1. & -0.1878 & -2.918 \\ 1.500 & -1.451 & -2.082 \\ 2. & -2.241 & -1.068 \\ 2.500 & -2.529 & -0.1092 \\ 3. & -2.384 & 0.6485 \end{array} \right] \right]$$