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, then numeric solutions using NDSolve. 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 studied using the phi function. This is a function defined using the formula for the solution to the generic initial conditions $y(0) = y0$ and $y'(0) = v0$. The solution formula is made into the function phi of the parameters y0 and v0, and time t:

$$y = \varphi(y0, v0, t)$$

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

$$v = \partial_t\, \varphi(y0, v0, t)$$

Example. Obtain the phi function for the following damped, unforced mass spring system

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

and use it to plot time series for y and v and trajectories in the phase plane

We will first generate the solution formula for the generic initial conditions. Note that we have asked for the y[t] solution.

```
DE1 = y''[t] + 0.2y'[t] + y[t] == 0;
soln1 = DSolve[ {DE1, y[0]==y0, y'[0]==v0}, y[t], t]
```

```
{{y[t] → e^(-0.1 t) (1. y0 Cos[0.994987 t] +
      1.00504 v0 Sin[0.994987 t] + 0.100504 y0 Sin[0.994987 t])}}
```

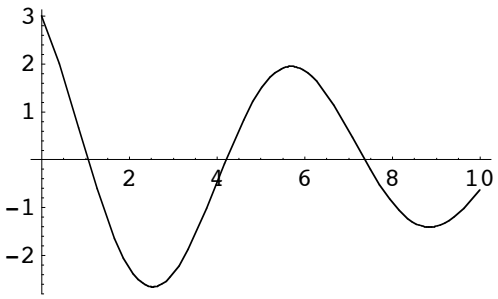
The output cell for soln1 contains the formula for the phi function. It can be defined using the following entry. We call is phi1 because we will be defining other phi functions later in this section.

```
phi1[y0_,v0_,t0_] := Evaluate[ y[t]/.soln1 ]
```

For example, the solution satisfying $y(0) = 3$ and $y'(0) = -2$ is given by $\varphi\,(3,-2,t\,)$

```
phi1[3,-2,t]
Plot[ phi1[3,-2,t], {t,0,10} ]
```

```
{e^(-0.1 t) (3. Cos[0.994987 t] - 1.70856 Sin[0.994987 t])}
```

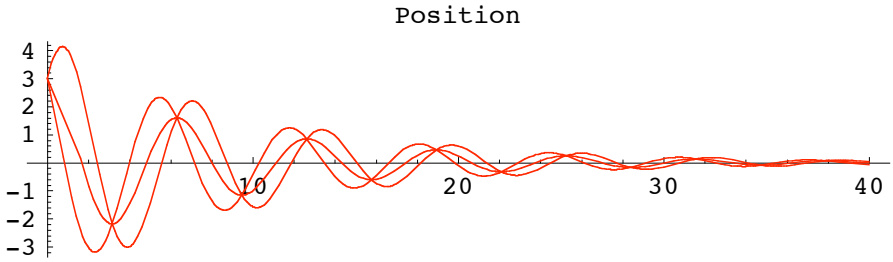


Noting that the system is underdamped with time constant 10 seconds, in about 40 seconds any solution that can graphed with reasonable scale factors will disappear from view. The first plot below is a family of three time series for position, using the initial conditions

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

Note the use Evaluate on the Table function.

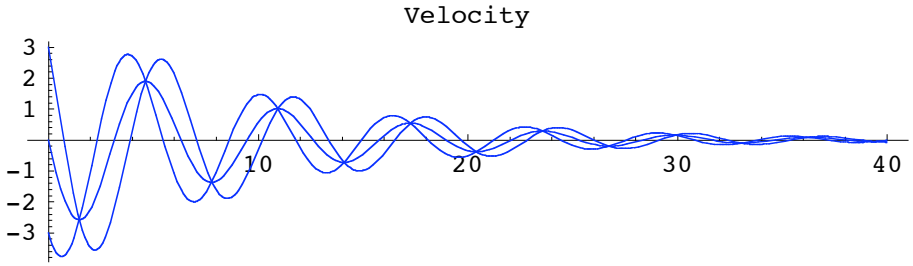
```
Plot[ Evaluate[Table[phi1[3,v,t],{v,-3,3,3}]], {t,0,40},
      PlotStyle->RGBColor[1,0,0], AspectRatio->1/4, PlotLabel->"Position" ]
```



Here are the corresponding velocity curves.

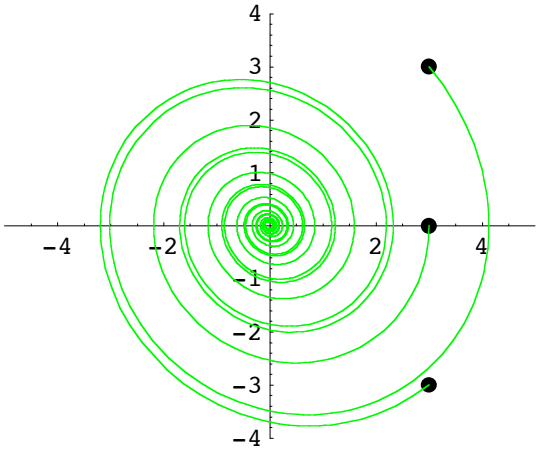
Recall that the partial derivative of phi with respect to t is obtained in *Mathematica* with D[phi[y0,t0,t] , t] .

```
Plot[ Evaluate[Table[D[phi1[3,v,t],t],{v,-3,3,3}]], {t,0,40},
      PlotStyle->RGBColor[0,0,1], AspectRatio->1/4, PlotLabel->"Velocity"
    ]
```



The phase plane trajectories are drawn next, in green. (Green? Sure, red + blue = green.) The three initial points have also been plotted.

```
inits = ListPlot[ Table[{3,k},{k,-3,3,3}], PlotStyle->PointSize[0.03]];
Show[ inits,
      ParametricPlot[
      Evaluate[Table[{phi1[3,v,t],D[phi1[3,v,t],t]},{v,-3,3,3}]],
              {t,0,40}, PlotStyle->RGBColor[0,1,0] ],
      PlotRange->{{-5,5},{-4,4}}, AspectRatio->4/5]
```



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 = y''[t] + 0.2y'[t] + y[t] == Cos[t];
soln2 = DSolve[ {DE2, y[0]==y0, y'[0]==v0}, y[t], t]//FullSimplify
```

$$\{\{y[t] \to e^{-0.1\,t}\,(((-6.27647\times10^{-16}+4.33334\times10^{-33}\,i)+(0.+0.\,i)\,v0+(1.+1.40174\times10^{-17}\,i)\,y0)\,\text{Cos}[0.994987\,t]+((-5.02519-7.14864\times10^{-18}\,i)+(1.00504+3.49569\times10^{-20}\,i)\,v0+(0.100504-6.93889\times10^{-19}\,i)\,y0)\,\text{Sin}[0.994987\,t]+e^{0.1\,t}\,((6.66134\times10^{-16}+8.88178\times10^{-16}\,i)\,\text{Cos}[1.\,t]+(5.-2.22045\times10^{-16}\,i)\,\text{Sin}[1.\,t]))\}\}$$

The solution formula is quite complicated. Applying FullSimplify helped a little bit. It appears that *Mathematica* is using an algorithm that uses the complex roots of the characteristic equation.

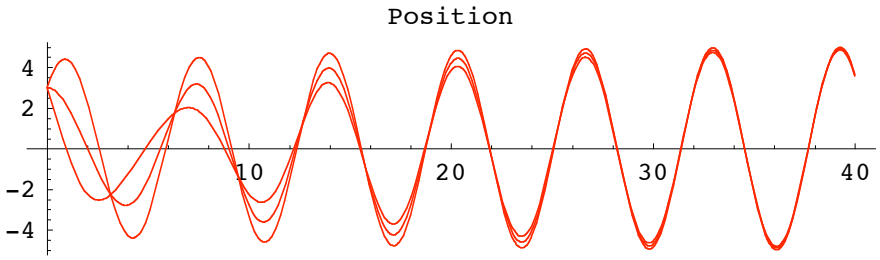
The new phi function.

```
phi2[y0_,v0_,t] := Evaluate[ y[t]/.soln2 ]
```

The time series for y with the same initial conditions:

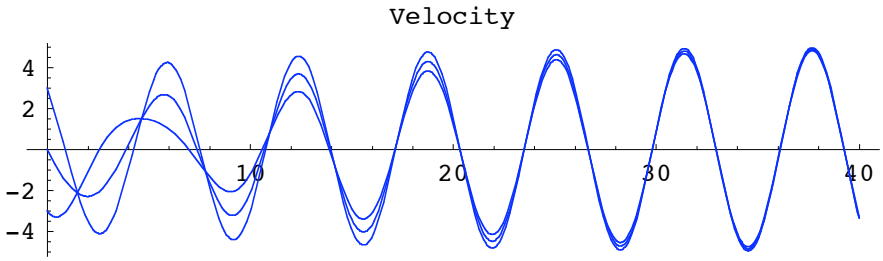
$$y(0) = 3,\ v(0) = -3,\ 0,\ 3\ .$$

```
Plot[ Evaluate[Table[phi2[3,v,t],{v,-3,3,3}]], {t,0,40},
      PlotStyle->RGBColor[1,0,0], AspectRatio->1/4, PlotLabel->"Position" ]
```



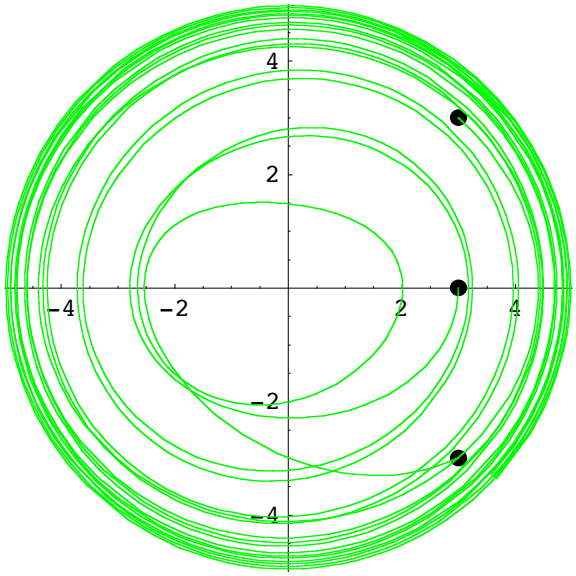
The velocity curves:

```
Plot[ Evaluate[Table[D[phi2[3,v,t],t],{v,-3,3,3}]], {t,0,40},
      PlotStyle->RGBColor[0,0,1], AspectRatio->1/4, PlotLabel->"Velocity"
    ]
```



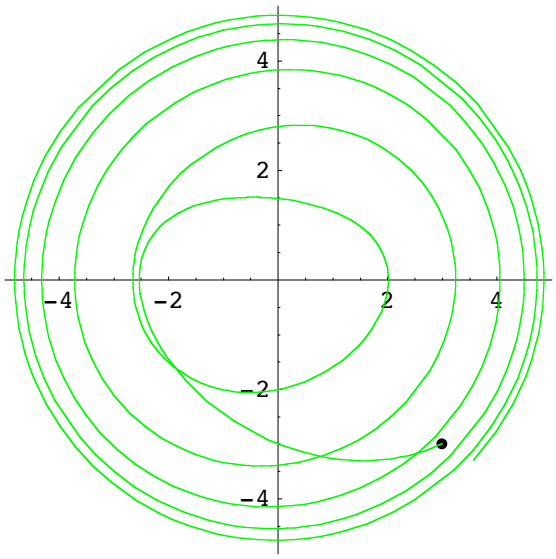
And the phase plane trajectories, all tangled up.

```
inits = ListPlot[ Table[{3,k},{k,-3,3,3}], PlotStyle->PointSize[0.03]];
Show[ inits,
      ParametricPlot[
      Evaluate[Table[{phi2[3,v,t],D[phi2[3,v,t],t]},{v,-3,3,3}]],
              {t,0,40}, PlotStyle->RGBColor[0,1,0] ],
      PlotRange->{{-5,5},{-5,5}}, AspectRatio→1/1]
```



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.

```
inits = ListPlot[ {{3,-3}}, PlotStyle->PointSize[0.02]];
Show[ inits,
      ParametricPlot[
      Evaluate[{phi2[3,-3,t],D[phi2[3,-3,t],t]}],
              {t,0,40}, PlotStyle->RGBColor[0,1,0] ],
      PlotRange->{{-5,5},{-5,5}}, AspectRatio→1/1]
```



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 ParametricPlot3D function

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 *Mathematica* using the ParametricPlot3D procedure. The syntax for a 3 dimensional curve is essentially the same as for the 3 dimensional curves we have been plotting with one exception:

The color specification for the curve must be included as the fourth entry in the coordinate list.

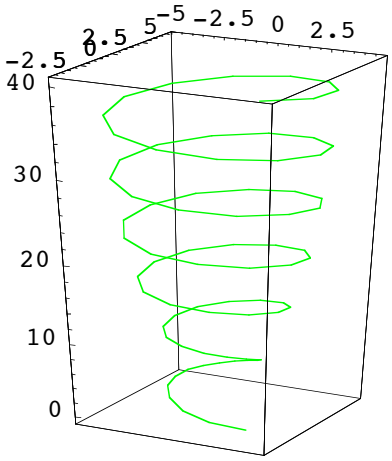
The last phase plane trajectory corresponds to the initial state

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

The state space trajectory is plotted below.

Note the application of Evaluate to the coordinate function.

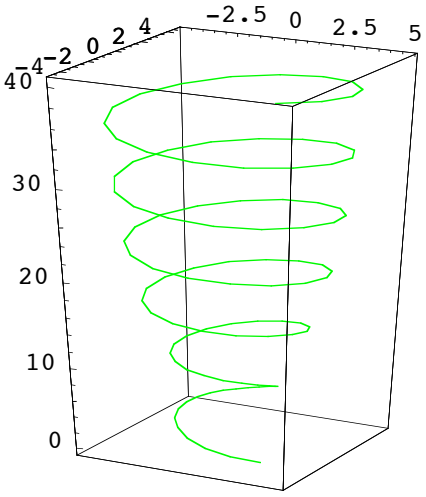
```
ParametricPlot3D[
       Evaluate[{phi2[3,-3,t],D[phi2[3,-3,t],t],t,RGBColor[0,1,0]}],
       {t,0,40}, AspectRatio->5/4 ]
```



As the default, ParametricPlot3d plots 70 points and choses to display the curve from the viewpoint of the spherical coordinate angles theta = 67.5, phi = 284.7 (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.) This can be determined by selecting the plot and choosing **Input/3D ViewPoint Selector...** . The view of the plot box also shows labels on the three axes.

The next entry adds the setting PlotPoints-> 100 to get a smoother curve.

```
ParametricPlot3D[
       Evaluate[{phi2[3,-3,t],D[phi2[3,-3,t],t,t,RGBColor[0,1,0]}],
       {t,0,40}, AspectRatio->5/4, PlotPoints→100 ]
```



Numerical solutions: NDSolve

The NDSolve function can be used to create phase plane pictures. If the equation is autonomous, tangent vectors can also be displayed using PlotVectorField. The display of several trajectories in the vector field is called a "flow".

The following entry shows how to create the vector field, F, associated with the differential equation named D1.

```
DE1
```

```
y[t] + 0.2 y′[t] + y′′[t] == 0
```

Note that it begins with two substitutions, the a solve, then another substitution.

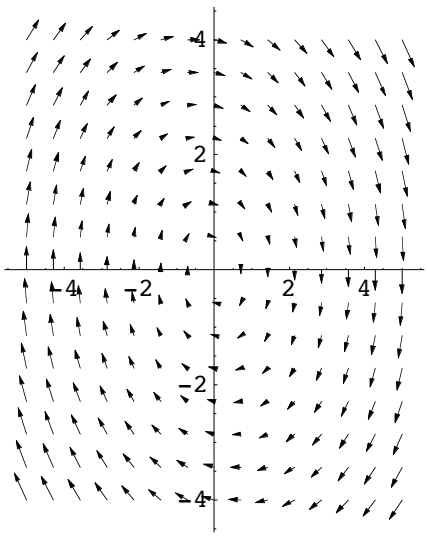
```
DE1/.{y'[t]->v,y[t]->y};
Solve[ %, {y''[t]} ];
F = {v,y''[t]}/.%[[1]]
```

```
{v, -0.2 v - 1. y}
```

And here is the plot of the vector field F.

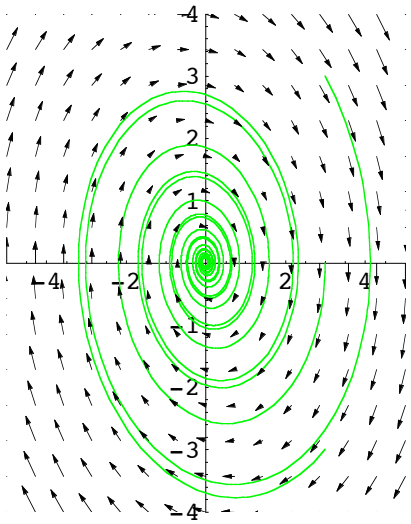
```
<<Graphics`PlotField`
```

```
vf1 = PlotVectorField[ F, {y,-5,5}, {v,-4,4}, Axes->True,
                       AspectRatio->5/4 ]
```



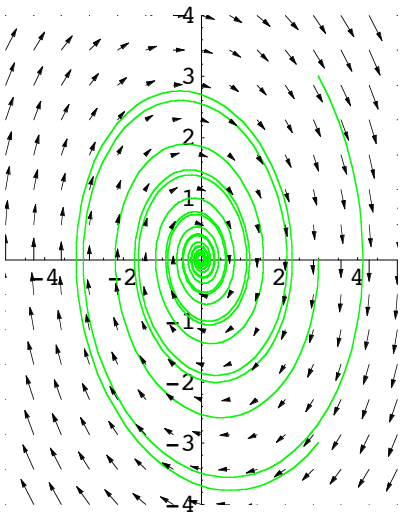
The next picture displays the vector field with several trajectories created with its phi function.

```
Show[
     ParametricPlot[
     Evaluate[Table[{phi[3,v,t],D[phi[3,v,t],t]},{v,-3,3,3}]], {t,0,40} ,
               PlotStyle->{RGBColor[0,1,0]} ],
     vf1, PlotRange->{{-5,5},{-4,4}}, AspectRatio->5/4 ]
```



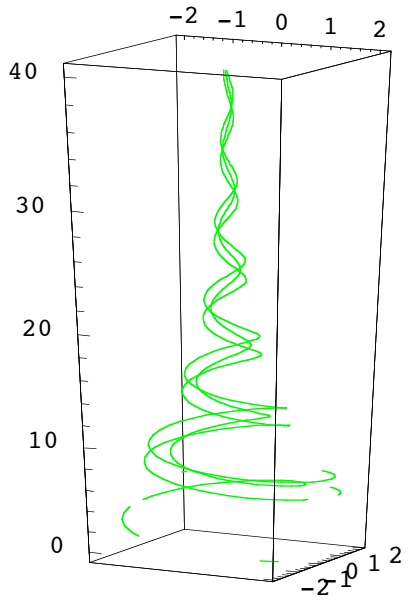
Here is the same picture created using NDSolve. The trajectories are plotted using the interpolation functions generated by NDSolve.

```
nsolns = Evaluate[ Table[
      NDSolve[{DE1,y[0]==3,y'[0]==v0}, y, {t,0,40}], {v0,-3,3,3}] ];
Show[
    ParametricPlot[
    Evaluate[Table[{y[t],y'[t]}/.nsolns[[k]],{k,3}]], {t,0,40} ,
             PlotStyle->{RGBColor[0,1,0]} ],
    vf1, PlotRange->{{-5,5},{-4,4}}, AspectRatio->5/4 ]
```



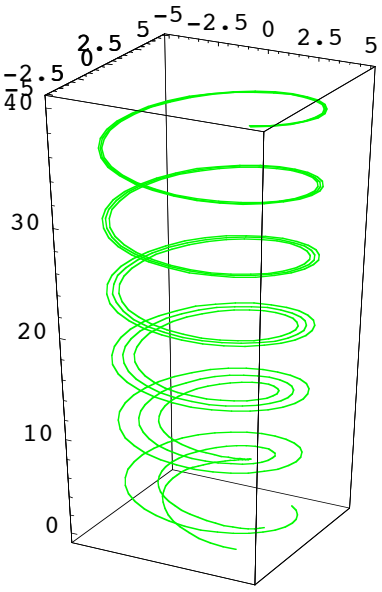
The following plot shows the state space trajectories for the same solutions.

```
ParametricPlot3D[
      Evaluate[Table[{y[t],y'[t],t,RGBColor[0,1,0]}/.nsolns[[k,1]],{k,3}]],
      {t,0,40} , PlotPoints->200, AspectRatio->5/3 ]
```



The state space trajectories for the family of forced (tangled) trajectories for DE2 are shown next. They are plotted as above, using the numerical solutions. These curves do not intersect in state space.

```
nsolns2 = Evaluate[ Table[
      NDSolve[{DE2,y[0]==3,y'[0]==v0}, y, {t,0,40}], {v0,-3,3,3}] ];
ParametricPlot3D[
      Evaluate[Table[{y[t],y'[t],t,RGBColor[0,1,0]}/.nsolns2[[k,1]],{k,3}]],
      {t,0,40} , PlotPoints->200, AspectRatio->5/3 ]
```



Numerical output

Finally, the following input shows how to get a nice display of numerical solution values. We use DE2 with the initial conditions $y(0) = 3$ and $y'(0) = -3$.

```
nsoln = NDSolve[ {DE2, y[0]==3, y'[0]==-3}, y, {t,0,5}]
```

```
{{y → InterpolatingFunction[{{0., 5.}}, <>]}}
```

```
Join[   {{t,y[t],y'[t]}},
        Evaluate[Table[{t,y[t],y'[t]},{t,0,5,0.5}]]/.nsoln[[1]]
    ]//MatrixForm
```

t	y[t]	y′[t]
0	3.	-3.
0.5	1.39187	-3.30319
1.	-0.187791	-2.91812
1.5	-1.45094	-2.08228
2.	-2.24065	-1.06843
2.5	-2.52896	-0.109156
3.	-2.38395	0.648519
3.5	-1.92326	1.15176
4.	-1.27159	1.41968
4.5	-0.534044	1.5046
5.	0.210605	1.45408