Part IV. Linear Differential Equations

Section 1. Linear Oscillators

Let's set the stage by recalling some of the things that you are learning in your differential equations course.

A first order linear ordinary differential equation has the general form

$$a(t)\,y'(t) + b(t)\,y(t) = f(t)$$

or

$$a\,y' + b\,y = f$$

for short. Some equations like these were solved and analyzed in Part II. See the differential equation that was used to model Juanita's credit card payments and the equation modeling the temperature inside of a house on a sunny day. A second order linear differential equation has the general form

$$a\,y'' + b\,y' + c\,y = f.$$

The letters a, b, c, and f represent known functions of, say, time, and $y = y(t)$ is unknown. This is called a linear oscillator equation when a, b, and c are constant functions, a and c positive, b non-negative. Certainly the most famous example of a linear oscillator is the model for an object of mass m as it oscillates on a spring with spring constant k and possibly slowed down by a force proportional to its speed. The constant of proportionality is called the damping constant and is often denoted b. Application of Newton's equally famous

$$Force = Mass \text{ x } Acceleration$$

yields the equation

$$m\,y'' + b\,y' + k\,y = f$$

The function f is called the forcing function or the input because it usually represents an external force applied to the mass. If $f = 0$, the equation is called "unforced" (or homogeneous in the general case). See Ledder, Chapter 3, Section 1.

Solve this too: dsolve *and the unforced equation*

Maple's **dsolve** procedure can be applied to an ordinary differential equation of any order. In particular it can supply the solution formula for any unforced linear oscillator equation. Most of this section is devoted to examples illustrating how to obtain solutions and plot their graphs.

Example An object of mass 4.3 kilograms is attached to a spring with spring constant 15.2 Newtons per meter. Find the general solution if the system is undamped.

```
> undamped_DE := 4.3*diff(y(t),t,t) + 15.2*y(t) = 0;
  gen_soln := dsolve( undamped_DE );
```

$$undamped\_DE := 4.3\left(\frac{d^2}{dt^2}y(t)\right) + 15.2\,y(t) = 0$$

$$gen\_soln := y(t) = \_C1\sin\left(\frac{2}{43}\sqrt{1634}\,t\right) + \_C2\cos\left(\frac{2}{43}\sqrt{1634}\,t\right)$$

Typical Problem Pull the mass 2 meters in the positive direction. Push it away from equilibrium with an

initial speed of 3 meters per second. Plot the solution. How far does the mass move in the positive direction? When does it pass through the equilibrium position for the first time? How fast is it going at that time?

This is an initial value problem, $y(0) = 2, y'(0) = 3$. These equations are entered into **dsolve**, along with the differential equation, inside of set brackets. Observe that initial conditions are entered in the form

```
y(0)=2, D(y)(0)=3 .
```

```
> Position := dsolve( {undamped_DE, y(0)=2, D(y)(0)=3} );
```

$$Position := y(t) = \frac{3}{76}\sin\left(\frac{2}{43}\sqrt{1634}\,t\right)\sqrt{1634} + 2\cos\left(\frac{2}{43}\sqrt{1634}\,t\right)$$

It is easier to get answers to the questions that have been asked when the solution formula is in function form. The **unapply** procedure is used to make the solution function, named g. At the same time the exact values in the solution formula are replaced with 4 digit approximations.

```
> g := unapply(evalf[4](rhs(Position)),t);
```

$$g := t \to 1.595\sin(1.880\,t) + 2.\cos(1.880\,t)$$

Having done this, all floating point computations should be made to 4 digit accuracy. It makes no sense to calculate to 10 digits when the constants are only accurate to 4. This can be done by reassigning a global Maple constant named **Digits** . (Its default value is 10.)

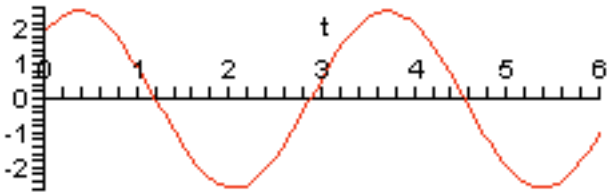
```
Digits := 4
```

```
> Digits := 4;
```

$$Digits := 4$$

The graph of the solution is a shifted sine wave or, if you prefer, a shifted cosine wave. Such curves are referred to as sinusoids.

```
> plot( g(t), t=0..6);
```



To find how far the mass moves in the positive direction we will find the first maximum value of g for t positive. Name the t value *tmax*. The number *ymax* is obtained as $g(tmax)$.

```
> tmax := fsolve( D(g)(t)=0, t, 0..1);
  ymax := g(tmax);
```

$$tmax := 0.3581$$

$$ymax := 2.558$$

So, the mass initially moves 0.558 meters in the positive direction to the position 2.558 meters from equilibrium.

To find when the mass passes through equilibrium, we solve the equation $g(t) = 0$. The first positive root of g

is between 0 and 2. We name it t0.

```
> t0 := fsolve( g(t)=0, t, 0..2);
```

$$t0 := 1.194$$

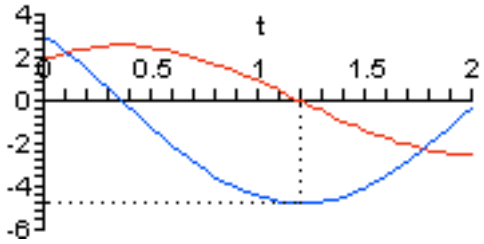
The velocity of the object at time t0 is D(g)(t0). It will be negative (see the graph), so apply the absolute value function **abs** to get the speed.

```
> speed = abs(D(g)(t0))*meters_per_second;
```

$$speed = 4.809\ meters\_per\_second$$

This information is plotted on the graph of the velocity function below (blue curve, dotted lines).

```
> DOTS := [ [0,-4.8],[t0,-4.8],[t0,0] ]:
  plot( [g(t),D(g)(t),DOTS], t=0..2, -6..4, color=[red,blue,black],
        linestyle=[SOLID$2,DOT]);
```



The phase plane trajectory

The information contained in the last plot can also be displayed using what is known as the phase plane trajectory. This is a plot of the parametrized curve $y = y(t)$, $v = y'(t)$ in two dimensional y,v space called the phase plane (v denotes velocity). Since g is defined so that $y = g(t)$, the phase plane trajectory for the first 2 seconds is obtained by applying the plot procedure to the following list.

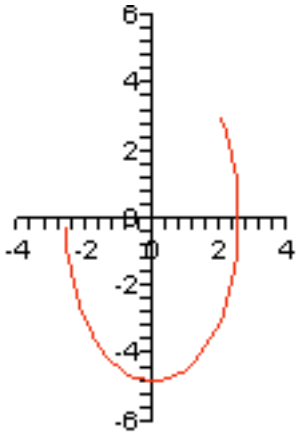
```
[g(t),D(g)(t),t=0..2]
```

The optional equation

```
scaling=constrained
```

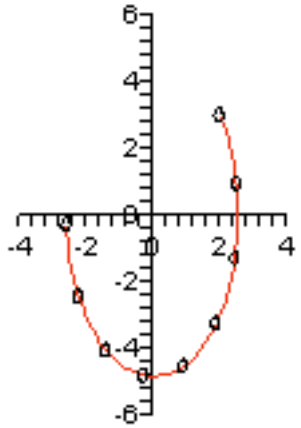
instructs the **plot** procedure to use the same scale on the two axes.

```
> plot( [g(t),D(g)(t),t=0..2], color=red, scaling=constrained,
        view=[-4..4,-6..6]);
```



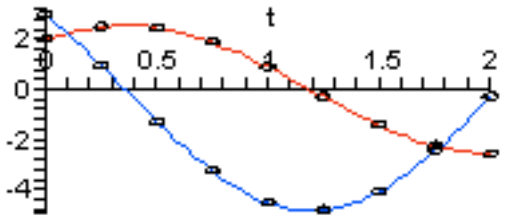
The next plot shows the same trajectory, but with the phase points corresponding to position and velocity at the 9 time values $t = 0, 0.25, 0.5, \ldots, 2.0$. The entries **symbol=circle** and **symbolsize=16** plots the points as circles of size 16 (points).

```
> Trajectory := [g(t),D(g)(t),t=0..2]:
  Pts := [ [g(t/4),D(g)(t/4)] $ t=0..8]:
  plot( [ Trajectory, Pts ], color=[red,black], style=[line,point],
          symbol=circle, symbolsize=16, scaling=constrained,
          view=[-4..4,-6..6]);
```



Unlock the key to phase plane trajectories . Given a point on the trajectory, its first coordinate is the position of the object and its second coordinate is its velocity. The nine points on the trajectory abov correspond to the nine pairs of points appearing on the position and velocity curves shown below. By a "pair" of points we mean two points corresponding to the same t value.

```
> P := [ [k/4,g(k/4)]$k=0..8]:
  V := [ [k/4,D(g)(k/4)]$k=0..8]:
  plot( [g(t), D(g)(t), P, V  ], t=0..2, color=[red,blue,black$2],
              style=[line$2,point$2], symbol=circle, symbolsize=16);
```



Simple Harmonic Motion

Simple harmonic motion is the name given to the motion of an undamped mass on a linear spring. The plot of position versus time, called a "time series", is a sinusoid. The phase plane trajectory for simple harmonic motion is an ellipse. Only a portion of the ellipse is shown above. The time it takes a phase point to traverse the ellipse is the same time it takes the mass spring system to return to its original configuration. The motion is periodic. The extreme values on the time series for position correspond to the points on the phase plane trajectory that are farthest from the vertical velocity axis. The extreme values on the time series for velocity correspond to the points on the phase plane trajectory that are farthest from the horizontal position axis.

Damped, unforced motion (underdamped)

We now examine the geometry of damped, unforced oscillations. Because the solutions still oscillate the system is called underdamped. The overdamped case is considered in the exercises. Consider the same mass and spring, but add a damping term with $b = 1.0$.

```
> damped_DE := 4.3*diff(y(t),t,t) + 1.0*diff(y(t),t) + 15.2*y(t) = 0;
  gen_soln := dsolve( damped_DE );
```

$$damped\_DE := 4.3\left(\frac{d^2}{dt^2}y(t)\right) + 1.0\left(\frac{d}{dt}y(t)\right) + 15.2\,y(t) = 0$$

$$gen\_soln := y(t) = \_C1\ \mathbf{e}^{\left(-\frac{5}{43}t\right)}\sin\left(\frac{1}{43}\sqrt{6511}\ t\right) + \_C2\ \mathbf{e}^{\left(-\frac{5}{43}t\right)}\cos\left(\frac{1}{43}\sqrt{6511}\ t\right)$$

Observe that the general solution still contains a sine and a cosine, but now a decaying exponential term appears on each one. Factor it out and what remains is a sinusoid.

• Maple's **collect** procedure can be used to factor a common term out of an expression. The syntax is

```
collect( expression, keyword)
```

where "keyword" identifies the type of term that is to be factored out, in this case the exponential term.

```
> collect(gen_soln,exp);
```

$$y(t) = \left(\_C1\sin\left(\frac{1}{43}\sqrt{6511}\ t\right) + \_C2\cos\left(\frac{1}{43}\sqrt{6511}\ t\right)\right)\mathbf{e}^{\left(-\frac{5}{43}t\right)}$$

Typical Problem Pull the mass 2 meters in the positive direction and push it away from equilibrium with an initial speed of 3 meters per second, just like before. Plot the solution and answer the same questions about its motion. Compare the answers to the ones obtained in the undamped case.

```
> Position := dsolve( {damped_DE, y(0)=2, D(y)(0)=3} );
```

$$Position := y(t) = \frac{139}{6511}\sqrt{6511}\ \mathbf{e}^{\left(-\frac{5}{43}t\right)} \sin\left(\frac{1}{43}\sqrt{6511}\ t\right) + 2\,\mathbf{e}^{\left(-\frac{5}{43}t\right)} \cos\left(\frac{1}{43}\sqrt{6511}\ t\right)$$

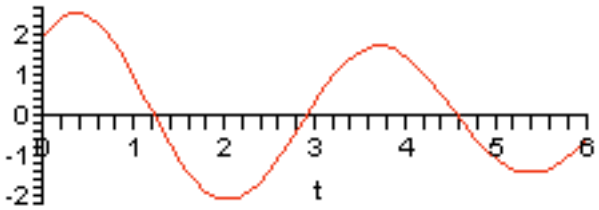
The position function for underdamped motion will be named G. Because Digits is still set to 4 there is no reason to use **evalf[4]** to get a 4 digit approximation for the constants in the formula for G.

```
> G := unapply(evalf(rhs(Position)),t);
```

$$G := t \rightarrow 1.723\ \mathbf{e}^{(-0.1163\,t)} \sin(1.877\,t) + 2.\ \mathbf{e}^{(-0.1163\,t)} \cos(1.877\,t)$$

The graph:

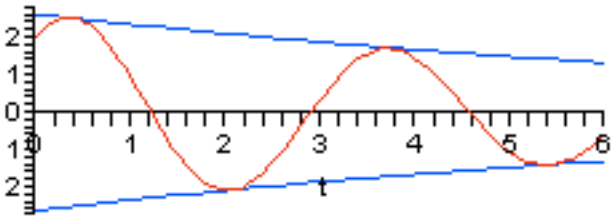
```
> plot( G(t), t=0..6);
```



This not a sinusoid because the oscillations are not of constant amplitude, they are decreasing. The decrease is called "exponential" because the amplitude decreases with an cxponential envelope. See the following picture where the expenontial envelope has been added to the plot.

```
> Envelope := exp(-0.1163*t)*sqrt(1.723^2 + 2^2);
  plot( [Envelope,-Envelope,G(t)], t=0..6, color=[blue$2,red] );
```

$$Envelope := 2.640\ \mathbf{e}^{(-0.1163\,t)}$$



The time constant for the exponential term is approximately 9 seconds (43/5) so, over the 6 second time interval shown above, the oscillations decrease by a factor of about exp(6/9) x 1/3 = 0.65. The t value for the widest oscillation will be called *tmaxG* the corresponding G value is named *ymaxG*.

```
> tmaxG := fsolve( D(G)(t)=0, t, 0..1);
  ymaxG := G(tmaxG);
```

$$tmaxG := 0.3459$$

$$ymaxG := 2.531$$

The damped mass moves 0.531 meters in the positive direction (compared to 0.558 meters before) and it gets

there a tiny bit faster (the *tmax* value calculated earlier was 0.3581). The time that the damped mass first passes through equilibrium will be denoted t0G.

```
> t0G := fsolve( G(t)=0, t, 0..2);
```

$$t0G := 1.216$$

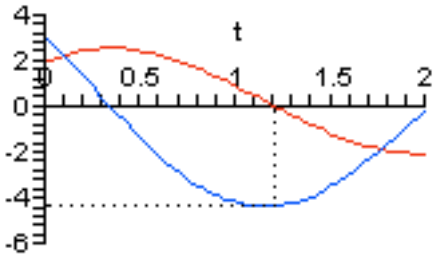
Recall that the undamped mass got to the equilibrium for the first time at $t = 1.194$ seconds. It was traveling at a speed of 4.809 meters a second. The next calculation shows that the damped mass not only took longer, but it is also going more slowly through equilibrium, traveling 4.301 meters per second.

```
> speedG = abs(D(G)(t0G))*meters_per_second;
```

$$speedG = 4.301\ meters\_per\_second$$

This information goes on the the graph of the velocity function, with dotted lines, just like before.

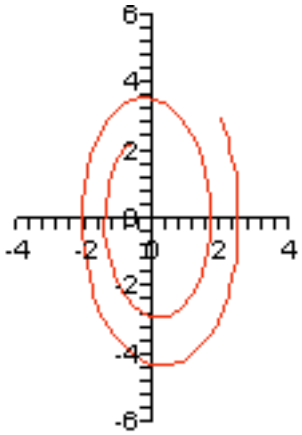
```
> DOTS := [ [0,-4.3],[t0G,-4.3],[t0G,0] ]:
  plot( [G(t),D(G)(t),DOTS], t=0..2, -6..4, color=[red,blue,black],
        linestyle=[SOLID$2,DOT]);
```



The phase plane trajectory for underdamped oscillations

The motion of the damped system is no longer periodic and the phase plane trajectory is no longer an ellipse. It spirals inward. See the plot below where the time of motion has been extended to 6 seconds.

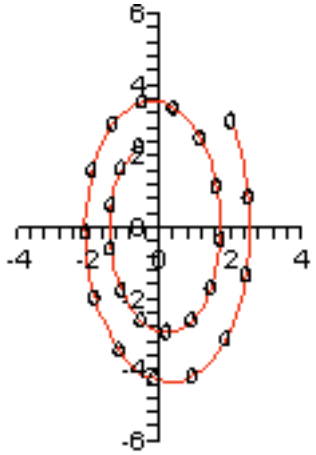
```
> plot( [G(t),D(G)(t),t=0..6], color=red, scaling=constrained,
        view=[-4..4,-6..6]);
```



The next picture adds the 25 points to the trajectory corresponding to position and velocity at time values $t =$ 0, 0.25, 0.5, ... , 6.0.

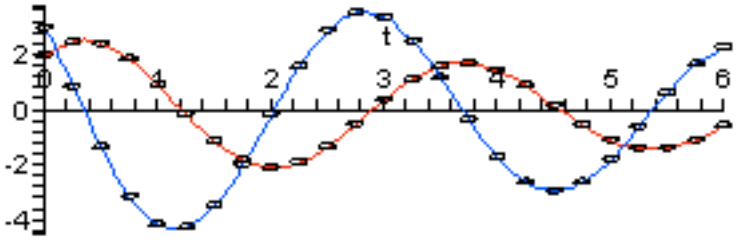
```
> Trajectory := [G(t),D(G)(t),t=0..6]:
```

```
Pts := [ [G(t/4),D(G)(t/4)] $ t=0..24]:
plot( [ Trajectory, Pts ], color=[red,black], style=[line,point],
        symbol=circle, symbolsize=16, scaling=constrained,
        view=[-4..4,-6..6]);
```



And here are the corresponding pairs of points on the time series graphs.

```
> PG := [ [k/4,G(k/4)]$k=0..24]:
  VG := [ [k/4,D(G)(k/4)]$k=0..24]:
  plot( [G(t), D(G)(t), PG, VG  ], t=0..6, color=[red,blue,black$2],
              style=[line$2,point$2], symbol=circle, symbolsize=12);
```



Phase plane trajectories and vector fields

When the phase plane trajectory passes through the point (y, v) its velocity vector, which is tangent to the trajectory, is $< y', v' >$. (Maple uses this notation for a column vector, so we will use it also). Since $y' = v$ and $v' = y''$, we have

$$< y', v' > = < v, y'' > .$$

This tells us several things about the trajectories for a linear oscillator.

1. All phase plane trajectories cross the y axis with a vertical tangent. This is because $v = 0$ on the y axis so the first component of the velocity vector is also 0.

2. Similarly, trajectories move to the right when they are in the upper half plane, and move to the left in the lower half plane.

3. Since the differential equation determines y'' in terms of y and y' (= v), the trajectory's velocity vector can be found without knowledge of the solution formula.

Consider, for example, a second order linear constant coefficient homogeneous equation: (a not zero)

$$a\,y'' + b\,y' + c\,y = 0.$$

Substitute $y' = v$ and solve for y'' to obtain

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

Consequently, when a phase plane trajectory is at $< y, v >$, its velocity vector is

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

This vector equation can be expressed equivalently as a system of two first order scalar differential equations.

$$y' = v$$

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

The differential equation is called autonomous because the two derivatives above do not depend directly upon the time variable. In this case, a plot of an array of tangent vectors in the y,v plane gives the same information about the trajectories associated with a second order differential equation as a direction field gives for a first order equation. Such an array is called the vector field associated with the differential equation.

Example. The vector field for the damped mass spring system in this section is defined below. We name it F.

```
> damped_DE;
```

$$4.3\left(\frac{d^2}{dt^2}y(t)\right) + 1.0\left(\frac{d}{dt}y(t)\right) + 15.2\,y(t) = 0$$

```
> F := <v,(-15.2*y-1.0*v)/4.3>;
```

$$F := \begin{bmatrix} v \\ -3.536\,y - 0.2326\,v \end{bmatrix}$$

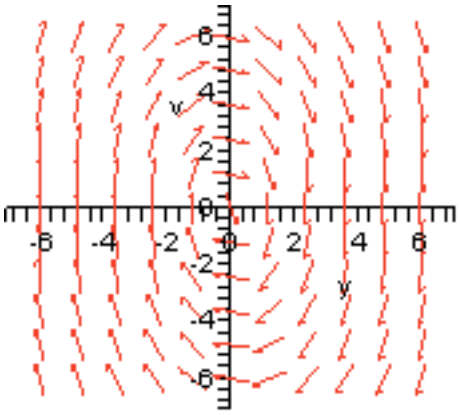
- By using arrow brackets < > to define F we are telling Maple that it is a Vector (capital V). Maple stores, displays, and computes with Vectors as columns. An entry of the form

  `V := < a | b >`

  would define V as a Vector stored, displayed, and computed in row form. More about this later.

Maple's **DEplot** procedure (**DEtools** package) will plot the vector field and solution trajectories. It creates the arrows directly from the differential equation, provided it is entered as a set containing the corresponding first order differential equations: $y' = v$ and $v' = -3.536\,y - 0.2326\,v$. The syntax is similar to what is used to apply **DEplot** to a single first order differential equation. See the next entry where 121 arrows (11 x 11) have been requested in the window $y = -6..6$, $v = -6..6$. The t-range, $t=0..6$, is required in the input, and must come right after the entry **[*y*(t), *v*(t)]** in the input sequence. See the **Help** page for **DEtools[DEplot]**.

```
> DEtools[DEplot]( {diff(y(t),t) = v(t),
                    diff(v(t),t) = -3.536*y(t) - 0.2326*v(t)},
                    [y(t),v(t)], t=0..6, y=-6..6, v=-6..6,
                    dirgrid=[11,11]);
```



Observe that the velocity vectors (the arrows) appear normalized to the same length. If initial values are entered as a set of lists

```
{ [y(t1)=y1,D(y)(t1)=v1], [y(t2)=y2,D(y)(t2)=v1], ... }
```

then **DEplot** draws the trajectories for the specified *t*-range. The default numerical algorithm is the classical Runge-Kutta method.

```
> DEtools[DEplot]( {diff(y(t),t) = v(t),
                    diff(v(t),t) = -3.536*y(t) - 0.2326*v(t)},
                    [y(t),v(t)], t=0..6, y=-6..6, v=-6..6,
                    dirgrid=[11,11], {[y(0)=2,v(0)=3]},
                    stepsize=0.1, linecolor=blue);
```