Motion Along A Straight Line, Part I: Position -> Velocity -> Acceleration

Introduction

OBJECTIVE: To apply special Maple functions to analyze position, velocity, and acceleration simultaneously.

Note: There are no executable commands in this Introduction section.

Structural engineers must design high-rise buildings to withstand earthquakes. When a high-rise building moves back and forth during an earthquake, the motion of any single floor is also back and forth along a straight, horizontal line. For earthquake design, the engineer must be able to completely describe the position, velocity, and acceleration of each of the floors as functions of time, and understand the relationships among these three functions. This provides the engineer with some of the information and understanding that is needed for the seismic design of the structure in a high-rise building. (In physics and engineering, the study of motion is called kinematics and falls in the realm of mechanics.)

This module includes three specially designed Maple commands that present dramatic animated visualizations of the derivative relations among the position, velocity, and acceleration. In addition to the seismic vibration of each floor in a building, a variety of other motions are investigated including constant velocity, constant acceleration, harmonic oscillation, and decaying oscillation. You can also use the specialized Maple commands to study other motions that are of interest to you.

To measure position along a line, we select a fixed point on the line as a reference point or origin, and we scale the line in appropriate units (e.g., meters, feet, or miles). A distance and a direction specify a position on the line. One direction from the reference point is taken as positive and the other direction as negative. Distance is measured using the units of scale along the line.

Before looking at the vibrations of a building shaken by an earthquake, let's consider some other motions.

This module uses three specially designed functions, **velocity(s , time_intreval, periodic), acceleration(s , time_intreval, periodic),**

posvelacc(s , time_intreval, periodic). You may need to expand the size of the graphic box to view the animation more clearly. To do this, click and drag one of the corners of the box.

NOTE: The special commands **velocity( )**, **acceleration( )**, and **posvelacc( )** are not built-in Maple functions and are only available in this module. The arguments for all three functions are: **s**, a position function; **time_interval**, the time interval over which the motion occurs; and **periodic**, a flag to indicate whether the motion is periodic or not. In the Parts that follow you will see examples of how these functions are used.

Technology Guidelines

NOTE: If you have just finished a worksheet, **restart** Maple before executing a new worksheet.
TO OPEN SECTIONS,
Click on the **PLUS** sign at the left hand side of the screen *or* select **Expand All Sections** from the **View** drop down menu.

TO STOP AN EXECUTION
Click on **STOP** button from the toolbar.

ORDER OF EXECUTION
Execute commands in the order given. Do not skip any Maple Input lines within a given worksheet

Alternatively, you can execute the entire worksheet by selecting the **Execute Worksheet** command from the **Edit** drop down menu.

SAVING WORKSHEETS.

You can save anytime to any directory you choose, and it is wise to save often.
EXPERIENCING MAJOR PROBLEMS
Save if appropriate, then shut down Maple and start it up again.

Special Functions

The code that follows defines the three special functions that are used in this worksheet. You do not have to understand the code to do this worksheet, however, you will need to execute it in order to define the special functions that are used later. You can execute the code by placing the cursor anywhere in the input cell and then pressing Enter. After you execute the code in the next cell, proceed to Part I of the worksheet.

```
> restart;
  with(plots): with(plottools):
  velocity := proc(fnc, var, periodic)
    local f, dfdt,d2fdt2, fvalues, dfdtvalues, d2fdt2values, start, dfdtmaxt22,
        dfdtmin333t22,fmaxt2222, fmint22, fmin, fmax, roots22, maxes, mines, dfdtmax,
        dfdtmin333, size, p1, p2, p3, t_a, t_o, t_f, t_ind, i, dfdt2maxt,dfdt2mint33,
        fmint2222, k, tend, ta, p2_a, p2_1, p2_2, p2_3,p2_4,p2_5, p2_6, p2_7, p1_1,p1_2,
        p1_3, p1_4, p1_5, p3_1, p3_2, p3_3, p3_4, p3_5, p4, dfdtmaxt, frame, numframes,


    t_ind := lhs(var);
    t_o := op(1, rhs(var));
    t_f := op(2, rhs(var));

    f := unapply(fnc, t);
    ft:=t->piecewise(t>=t_o and t<=t_f, f(t));
    dfdt := unapply(diff(f(t), t), t);
    d2fdt2 := unapply(diff(dfdt(t), t), t);

    [seq(evalf(f(i*(t_f-t_o)/500)),i=1..500)];
  ### WARNING: comparisons against undefined should use type(... , undefined)
```

```
  fvalues := remove( x -> type( x , undefined ),%);
  [seq(evalf(dfdt(i*(t_f-t_o)/500)),i=1..500)];
### WARNING: comparisons against undefined should use type(... , undefined)
  dfdtvalues := remove( x -> type( x , undefined ),%);
  [seq(evalf(d2fdt2(i*(t_f-t_o)/500)),i=1..500)]:
### WARNING: comparisons against undefined should use type(... , undefined)
  d2fdt2values := remove( x -> type( x , undefined ),%);

  dfdtmaxt22 := max(op(dfdtvalues));
  dfdtmin333t22 := min(op(dfdtvalues));
  dfdt2maxt := max(op(d2fdt2values22));
  dfdt2mint33 := min(op(d2fdt2values22));

  fmaxt2222 := max(op(fvalues));
  fmint2222 := min(op(fvalues));
  fmin := fmint2222-max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  fmax := fmaxt2222+max(abs(dfdtmin333t22),abs(dfdtmaxt22));

  if periodic=1 then
    tend := evalf(t_f-1/25*(t_f-t_o),20);
  else
    tend := evalf(t_f,20);
  fi;

frame:=0;
numframes:=27;
inc:= evalf((t_f-t_o)/25.,20);

if evalf(0.05*(t_f-t_o))>=1 then tlow:=evalf(t_o-0.05*(t_f-t_o)); thigh:=evalf(t_f+0.05*(t_f-t

#####################################################################3

  for ta from t_o to tend by inc do

frame:=frame+1;

  p2_1 := plot(evalf(ft(t)),t=tlow..thigh, f=fmin..fmax, color=COLOR(RGB,0,0,1),
       linestyle=[1,3], scaling=CONSTRAINED, title="position vs time",view=[tlow..thigh,


### WARNING: comparisons against undefined should use type(... , undefined)
  if evalf(dfdt(ta)) <> `undefined` then

    #p2_2 := plots[textplot]([ta+.5, f(ta), "1"], color=COLOR(RGB,0,0,1));


l:=t->f(ta)+dfdt(ta)*(t-ta);

if evalf(dfdt(ta)) = 0
then lowlim:= t_o-1 else if evalf(dfdt(ta)) > 0    then lowlim:=max(t_o-1,evalf((fmin-f(ta))/df
```

```
   else lowlim:=max(t_o-1,evalf((fmax-f(ta))/dfdt(ta)+ta));
fi;
fi;

if evalf(dfdt(ta)) = 0
then uplim:=t_f+1 else if evalf(dfdt(ta)) > 0    then
uplim:=min(t_f+1,evalf((fmax-f(ta))/dfdt(ta)+ta));
   else uplim:=min(t_f+1,evalf((fmin-f(ta))/dfdt(ta)+ta));
fi;
fi;


    p2_a := plot(l(t), t=lowlim..uplim,
          l=fmin..fmax,color=gold,thickness=2, scaling=CONSTRAINED,
title="position vs time",view=[tlow..thigh, fmin..fmax]);


    if evalf(dfdt(ta)) >= 0 then
      p2_3 := pointplot([[ta, f(ta)+dfdt(ta)],[ta+1, f(ta)+dfdt(ta)],
           [ta+1,f(ta)],[ta,f(ta)]], color=COLOR(RGB,0,0,0),style=LINE);
    else
      p2_3 := pointplot([[ta, f(ta)], [ta+1, f(ta)],
           [ta+1, f(ta)+dfdt(ta)], [ta, f(ta)+dfdt(ta)]],
           color=COLOR(RGB,0,0,0), style=LINE);
    fi;

    if evalf(dfdt(ta)) <> 0 then

      #p2_4 := textplot([ta-0.25, f(ta)+0.5*dfdt(ta), "v"],
       #       color=COLOR(RGB,1,0,0));
      p2_5 := plottools[arrow]([ta, f(ta)], [ta,f(ta)+dfdt(ta)], .025*(t_f-t_o),

    else

      p2_4 := NULL;
      p2_5 := NULL;
    fi;

  else

    p2_2 := NULL;
    p2_3 := NULL;
    p2_4 := NULL;
    p2_5 := NULL;
    p2_a := NULL;
  fi;

  p2_6 := plot([fmin,fmax],t=1.001*t_o..1.002, color=white);

  p2_7 := plot(0, t=tlow..thigh, color=black);
```

```
 p2[frame] := display(p2_7, p2_6, p2_1, p2_a, p2_3, p2_5,   view=[tlow..thigh, fmin..fmax]

#########################################################################
 dfdtmax:=fmax-((fmin+fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);
 dfdtmin333:=fmin-((fmin+fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);

 if ta <> t_o then p1_1 := plot(dfdt(t),t=t_o..ta, discont=true, color=red); else p1_1:=NULL

### WARNING: comparisons against undefined should use type(... , undefined)
 if evalf(dfdt(ta)) <> `undefined` then
   #p1_4 := plots[textplot]([ta-0.25, f(ta)+0.5*dfdt(ta), "v"],
    #     color=COLOR(RGB,1,0,0));

   if evalf(dfdt(ta)) <> 0 then
     p1_2 := plottools[arrow]([ta, 0],[ta,dfdt(ta)], .025*(t_f-t_o), .1*(t_f-t_o), .25,
         color=COLOR(RGB,1,0,0));
   else
     p1_2 :=NULL;
   fi;

 else
   p1_4:=NULL;
   p1_2:=NULL;
 fi;

 p1_3 := listplot([dfdtmin333,dfdtmax], color=white);

 p1_5 := plot(dfdt(t)+1, t=(t_o-(t_f-t_o)/10)..(t_f+(t_f-t_o)/10), color=white,discont=true);

 p1[frame] := display( p1_2, p1_3, p1_5, p1_1, title="velocity vs time");

 ########################################################################

### WARNING: comparisons against undefined should use type(... , undefined)
 if evalf(dfdt(ta)) <> `undefined` then
   #p3_4 := plots[textplot]([.3, f(ta)+0.5*dfdt(ta), "v"],color=COLOR(RGB,1,0,0));

   if evalf(dfdt(ta)) <> 0 then

   p3_1 := plottools[arrow]([0, f(ta)],[0, f(ta)+dfdt(ta)], .1, .4, .2,
         color=COLOR(RGB,1,0,0));
   else
   p3_1:=NULL;
   fi;

 else
   p3_1:=NULL;
   p3_4:=NULL;
 fi;

 p3_2 := pointplot([0,f(ta)], symbol=CIRCLE, axes=none);
```

```
  p3_3 := listplot([fmin, fmax], color=white);
  p3_5 := plot(0, t=-2..2, color=black, title="object's motion");

  p3[frame] := display(p3_1, p3_2, p3_3,  p3_5 );

##############################################################################

 p4[frame]:=display(display([p2[frame],p1[frame],p3[frame]],insequence=true), insequenc

 od;

display(seq(p4[aa], aa=1..frame), insequence=true);

end:
##############################################################################
acceleration := proc(fnc, var, periodic)

local f, dfdt,d2fdt2, fvalues, dfdtvalues, d2fdt2values, start, dfdtmaxt22,
      dfdtmin333t22,fmaxt2222, fmint22, fmin, fmax, roots22, maxes, mines, dfdtmax,
      dfdtmin333, size, p1, p2, p3, t_a, t_o, t_f, t_ind, i, dfdt2maxt,dfdt2mint33,
      fmint2222, k, tend, ta, p2_a, p2_1, p2_2, p2_3, p2_4, p2_5, p2_6, p2_7, p1_1,p1_2,
      p1_3,p1_4, p1_5, p3_1, p3_2, p3_3, p3_4, p3_5, p4, dfdtmaxt, ffvalues, fmaxt22,
      ffmax22, ffmin22, ff, q, frame, numframes, inc, p3aux,l,lowlim,uplim,ft,tlow,thigh ;

t_ind := lhs(var);
t_o := op(1, rhs(var));
t_f := op(2, rhs(var));



f:=unapply(diff(fnc,t),t);
ft:=t->f(t);
###ft:=t->piecewise(t>=t_o and t<=t_f,f(t));
dfdt:=unapply(diff(f(t),t),t);
ff:=unapply(fnc,t);

  [seq(evalf(f(i*(t_f-t_o)/500)),i=1..500)];
### WARNING: comparisons against undefined should use type(... , undefined)
  fvalues :=remove( x -> type( x , undefined ),%);
  [seq(evalf(dfdt(i*(t_f-t_o)/500)),i=1..500)]:
### WARNING: comparisons against undefined should use type(... , undefined)
  dfdtvalues := remove( x -> type( x , undefined ),%):
  [seq(evalf(ff(i*(t_f-t_o)/500)),i=1..500)]:
### WARNING: comparisons against undefined should use type(... , undefined)
  ffvalues := remove( x -> type( x , undefined ),%):

  dfdtmaxt22 := max(op(dfdtvalues));
  dfdtmin333t22 := min(op(dfdtvalues));
  dfdt2maxt := max(op(ffvalues));
```

```
  fmaxt22 := max(op(fvalues));
  fmint22 := min(op(fvalues));
  fmin := fmint22-max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  fmax := fmaxt22+max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  ffmax22:=max(op(ffvalues))+max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  ffmin22:=min(op(ffvalues))- max(abs(dfdtmin333t22),abs(dfdtmaxt22));

  if periodic=1 then
    tend := evalf(t_f-1/25*(t_f-t_o),20);
  else
    tend := evalf(t_f,20);
  fi;

frame:=0;
numframes:=27;
inc:= evalf((t_f-t_o)/25.,20);

if evalf(0.05*(t_f-t_o))>=1 then tlow:=evalf(t_o-0.05*(t_f-t_o)); thigh:=evalf(t_f+0.05*(t_f-t

#####################################################################3

  for ta from t_o to tend by inc do

frame:=frame+1;

  p2_1 := plot(evalf(ft(t)),t=t_o..t_f, f=fmin..fmax, color=COLOR(RGB,1,0,0),
       linestyle=[1,3], title="velocity vs time",view=[tlow..thigh, fmin..fmax],discont=true);



### WARNING: comparisons against undefined should use type(... , undefined)
  if evalf(f(ta)) <> `undefined` then

l:=t->f(ta)+dfdt(ta)*(t-ta);

if evalf(dfdt(ta)) = 0
then lowlim:=tlow else if evalf(dfdt(ta)) > 0    then lowlim:=max(tlow,evalf((fmin-f(ta))/dfdt
   else lowlim:=max(tlow,evalf((fmax-f(ta))/dfdt(ta)+ta));
fi;
fi;

if evalf(dfdt(ta)) = 0
then uplim:=thigh else if evalf(dfdt(ta)) > 0    then
uplim:=min(thigh,evalf((fmax-f(ta))/dfdt(ta)+ta));
   else uplim:=min(thigh,evalf((fmin-f(ta))/dfdt(ta)+ta));
fi;
fi;


    p2_a := plot(l(t), t=lowlim..uplim,
         l=fmin..fmax, color=gold,
```

```
        thickness=2, scaling=CONSTRAINED, title="velocity vs time",view=[tlow..thigh, fn

    #p2_2 := textplot([ta+.5, f(ta), "1"], color=COLOR(RGB,0,0,1));

   # p2_4 := textplot([ta-0.25, f(ta)+0.5*dfdt(ta), "v"],
    #       color=COLOR(RGB,1,0,0));

    if evalf(f(ta)) >= 0 then
      p2_3 := pointplot([[ta, f(ta)+dfdt(ta)],[ta+1, f(ta)+dfdt(ta)],
              [ta+1,f(ta)],[ta,f(ta)]],
              color=COLOR(RGB,0,0,0),
              style=LINE
              );
    else
      p2_3 := pointplot([[ta, f(ta)], [ta+1, f(ta)],
              [ta+1, f(ta)+dfdt(ta)], [ta, f(ta)+dfdt(ta)]],
              color=COLOR(RGB,0,0,0),style=LINE);
    fi;

    if evalf(dfdt(ta)) <> 0 then
      p2_5 := plottools[arrow]([ta, f(ta)], [ta,f(ta)+dfdt(ta)], .025*(t_f-t_o), .1*(t_f-t_o), .25,
              color=COLOR(RGB,0,1,0));
    else
      p2_5 := NULL;
    fi;

  else
    p2_a := NULL;
    p2_2 := NULL;
    p2_3 := NULL;
    p2_4 := NULL;
    p2_5 := NULL;
  fi;

  p2_6 := plot([fmin,fmax],t=1.001*t_o..1.001, color=white);


  p2_7 := plot(0, t=tlow..thigh, color=black);

  p2[frame] := display(p2_7, p2_6, p2_1, p2_3,  p2_5, p2_a,view=[tlow..thigh, fmin..fmax]);

##########################################################################

 dfdtmaxt:=fmax-((fmin/fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);
 dfdtmin333:=fmin-((fmin/fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);

 if t_o <> ta then p1_1 := plot(dfdt(t), t=t_o..ta, title="acceleration vs time", color=green, c

### WARNING: comparisons against undefined should use type(... , undefined)
 if evalf(f(ta)) <> `undefined` then
   if evalf(dfdt(ta)) <> 0 then
```

```
    p1_2 := plottools[arrow]([ta, 0],[ta,dfdt(ta)], .025*(t_f-t_o), .1*(t_f-t_o), .25,
            color=COLOR(RGB,0,1,0));
  else
    p1_2:=NULL;
  fi;

  #p1_3 := textplot([ta-0.25, 0.5*dfdt(ta), "a"],
  #       color=COLOR(RGB,1,0,0));
else
  p1_2:=NULL;
  p1_3:=NULL;
fi;

p1_4 := plot(dfdt(t)+1,  t=(t_o-(t_f-t_o)/10)..(t_f+(t_f-t_o)/10), color=white, discont=true);

p1_5 := plot([fmin,fmax], t=1.001*t_o..1.002, color=white);

p1[frame] := display(p1_2 , p1_4, p1_5, p1_1);

#############################################################################

### WARNING: comparisons against undefined should use type(... , undefined)
if evalf(f(ta)) <> `undefined` then

  if evalf(f(ta)) <> 0 then
    p3_1 := plottools[arrow]([0, ff(ta)],[0, ff(ta)+dfdt(ta)], .1, .4, .2,
            color=COLOR(RGB,0,1,0));
  else
    p3_1:=NULL;
  fi;

  p3_2 := pointplot([0,ff(ta)], symbol=CIRCLE);

  #p3_3 := textplot([.3, ff(ta)+0.5*dfdt(ta), "a"],color=COLOR(RGB,1,0,0));

else
  p3_1:= NULL;
  p3_2:= NULL;
  p3_3:= NULL;
fi;

p3_4 := listplot([ffmin22,ffmax22], color=white);
p3_5 := plot(0, -2..2, color=black, title="object's motion");

p3[frame] := display(p3_1, p3_2 , p3_4, p3_5, axes=none);

p3aux:= plottools[arrow]([0, ff(ta)],[0, ff(ta)+dfdt(ta)], .1, .4, .2,
            color=COLOR(RGB,0,1,0));

p3[1]:= display(p3aux, p3_2 , p3_4, p3_5, axes=none);
```

```
 p4[frame]:=display(display([p2[frame],p1[frame],p3[frame]],insequence=true), insequenc

#########################################################################
od;

 display(seq(p4[q],q=1..frame), insequence=true);


end:

#########################################################################

posvelacc:=proc(fnc, var, periodic)

local t_ind, t_o, t_f, f, dfdt, ff, fvalues, dfdtvalues, ffvalues22, dfdtmaxt22,
     dfdtmin333t22, dfdt2maxt, fmaxt22, fmint22, fmin, fmax, ffmax22, ffmin22, tend, ta,
     dfdtmax, dfdtmin333, p3_1, p3_2, p3_3, p3_4, p3_p, p3, i, ar, frame, numframes, inc;


t_ind := lhs(var);
t_o := op(1, rhs(var));
t_f := op(2, rhs(var));

f:=unapply(diff(fnc, t),t);
dfdt:=unapply(diff(f(t),t),t);
ff:=unapply(fnc,t);

if periodic=1 then
     tend := evalf(t_f-1/25*(t_f-t_o),20);
  else
     tend := evalf(t_f,20);
  fi;

frame:=0;
numframes:=27;
inc:= evalf((t_f-t_o)/25.,20);

#########################################################################3

  for ta from t_o to tend by inc do

frame:=frame+1;

  [seq(evalf(f(i*(t_f-t_o)/500)),i=1..500)]:
### WARNING: comparisons against undefined should use type(... , undefined)
  fvalues := remove( x -> type( x , undefined ),%):
  [seq(evalf(dfdt(i*(t_f-t_o)/500)),i=1..500)]:
### WARNING: comparisons against undefined should use type(... , undefined)
  dfdtvalues := remove( x -> type( x , undefined ),%):
  [seq(evalf(ff(i*(t_f-t_o)/500)),i=1..500)]:
```

```
### WARNING: comparisons against undefined should use type(... , undefined)
  ffvalues22 := remove( x -> type( x , undefined ),%):

  dfdtmaxt22 := max(op(dfdtvalues));
  dfdtmin333t22 := min(op(dfdtvalues));
  dfdt2maxt := max(op(ffvalues));

  fmaxt22 := max(op(fvalues));
  fmint22 := min(op(fvalues));
  fmin := fmint22-max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  fmax := fmaxt22+max(abs(dfdtmin333t22),abs(dfdtmaxt22));
  ffmax22:=max(op(ffvalues22))+max(abs(dfdtmin333t22),abs(dfdtmaxt22), abs(fmint22),
        abs(fmaxt22));
  ffmin22:=min(op(ffvalues22))- max(abs(dfdtmin333t22),abs(dfdtmaxt22), abs(fmint22),
        abs(fmaxt22));

  dfdtmax:=fmax-((fmin+fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);
  dfdtmin333:=fmin-((fmin+fmax)/2.0-(dfdtmin333t22+dfdtmaxt22)/2.0);



  ar:=ffmax22-ffmin22;




    p3_4 := plots[pointplot]([0,ff(ta)], symbol=CIRCLE, symbol=CIRCLE);




    if evalf(dfdt(ta))<>0 then p3_1 := plottools[arrow]([-0.02*ar-ar/20.0,ff(ta)],
          [-0.02*ar-ar/20.0,ff(ta)+dfdt(ta)],
          ar/80.0, ar/20, .2,
           color=COLOR(RGB,0,1,0)); else p3_1 :=NULL;
    fi;

    if evalf(ff(ta)) <> 0 then p3_2 := plottools[arrow]([0,0],[0,ff(ta)],
          ar/80.0, ar/20, .2, color=COLOR(RGB,0,0,1)); else p3_2 :=NULL;
    fi;

    if evalf(f(ta)) <> 0 then p3_3 := plottools[arrow]([0.02*ar+ar/20.0,ff(ta)],
          [0.02*ar+ar/20.0,ff(ta)+f(ta)],
          ar/80.0, ar/20, .2, color=COLOR(RGB,1,0,0)); else p3_3 :=NULL;
    fi;

    p3_4 := plots[pointplot]([0,ff(ta)], symbol=CIRCLE);


  p3_p := plots[pointplot]([0,f(ta)], symbol=circle);
```

```
    p3[frame] := plots[display](p3_1,p3_2,p3_3,p3_4,
            view=[-ar/2..ar/2, ffmin22..ffmax22]);

  od:

print(`s(blue), v(red) and a(green) vectors`);
print(plots[display]([seq(p3[i],i=1..frame)],insequence=true,
    view=[-ar/2..ar/2, ffmin22..ffmax22], axes=normal, tickmarks=[0,6]));

end:
```

Warning, the name changecoords has been redefined

Warning, the assigned name arrow now has a global binding

Part I: Constant Velocity

Chapter 3, Section 3

Suppose that you drive your car along a straight road at a constant speed of 60 mph for 4 hours and then turn around and come back at the same speed. Let's take the reference point to be your starting position, and the direction your car travels during the first four hours as the positive direction. The position of your car during the trip is given by the following function of time.

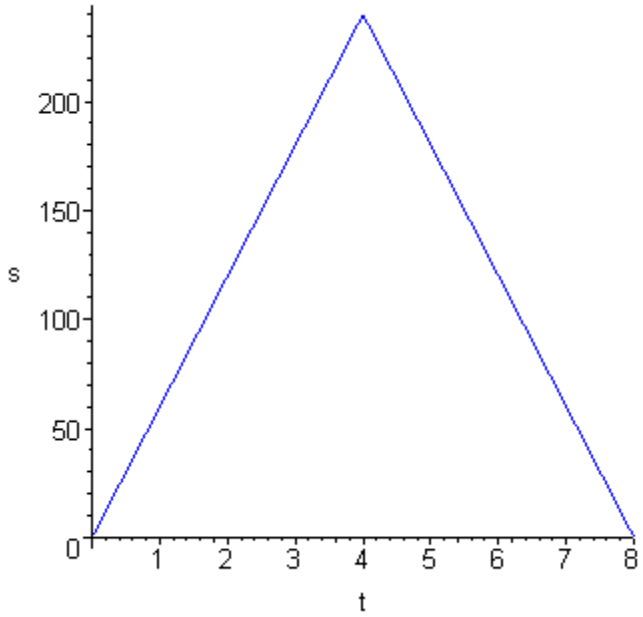
```
> s := piecewise(t <= 4, 60*t, t > 4, 480 - 60*t);
```

$$s := \begin{cases} 60\,t & t \le 4 \\ 480 - 60\,t & 4 < t \end{cases}$$

Note that in mechanics, the symbol **s** usually designates position.

Now we plot the function for the duration of the trip.

```
> plot(s, t=0..8, labels=["t","s"], color=blue);
```

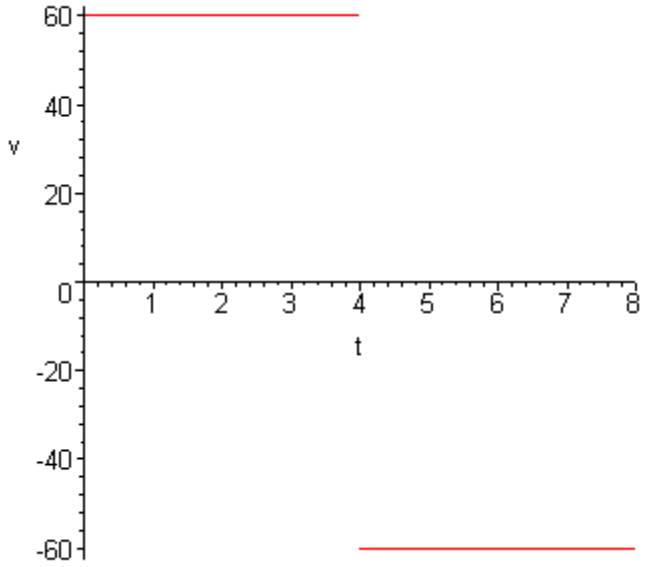


The velocity is defined to be the time rate of change of position. That is $v = \frac{ds}{dt}$. Let's find the velocity function and plot it.

> **v:=diff(s,t);**

$$v := \begin{cases} 60 & t < 4 \\ undefined & t = 4 \\ -60 & 4 < t \end{cases}$$

> **plot(v, t=0..8, labels=["t","v"], discont=true, color=red);**



According to Maple, the velocity, which is the derivative of the position function, is undefined at t = 4. Is that correct? To answer this question, you should look at the two one-sided derivatives of s at t = 4.

Like position, velocity has a size or magnitude and a direction. The magnitude or absolute value of an object's velocity is called its speed. If the position function, s(t), is increasing with time, then the velocity is positive, and if it is decreasing then the velocity is negative. Physical quantities that have a magnitude and a direction are called vectors, and the position and velocity of a moving object are vector quantities.

Recall that the derivative of a function at time t is the slope of the line tangent to the graph of the function at that point. Therefore, the velocity at time t is the slope of the position function at that time. The **velocity( )** command in the next executable section below illustrates this idea by producing a sequence of graphs that show: 1) the position function, s(t); 2) the velocity function, v(t); and 3) the object as it moves along a straight line. The velocity vector is shown on all three graphs. In the position-versus-time graph, the width of the small black rectangle is always one so that the length of the arrow is the slope of the gold tangent line. Note that the value for the last argument is zero since the motion is not periodic. In the first two animation graphs, the horizontal axis represents time.

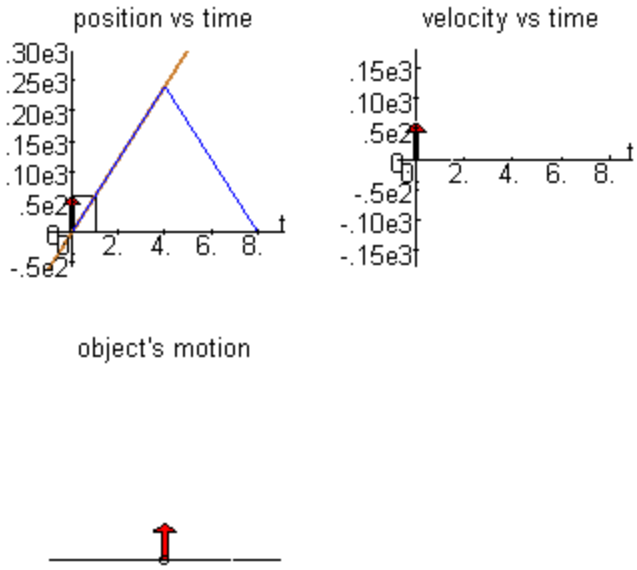
To animate the sequence of graphs:

1) Click on the graphic box that is produced by the next command and the animation toolbar will appear at the top of the screen.
2) Click "Play" button (**large single arrow**) to play the animation.

3) Click on the left or right **double arrow** buttons to speed up or slow down the animation.
4) Click on the **solid square** stop button to stop the animation.
5) You can also view the animation frame by frame by clicking the appropriate **right arrow-bar** button.

The next command may take a little while to execute, depending on the memory size and speed of your computer.

> **velocity(s, t=0..8, 0);**



A note about modeling: In the preceding model, the velocity is undefined at t = 4 hours. In reality, however, your car would have to slow down, turn around, and accelerate back to cruising speed at t = 4 hours. This maneuver would occur over a very short interval of time (say a few minutes) when compared to the 4-hour duration of the trip. The sharp corner at the turnaround time would in actuality be rounded, and the sudden jump from 60 mph to a mph would have a finite negative slope to it. On the scale of eight hours, however, the graphs would look much like those depicted above, even if the more precise model where used. With these observations in mind, the above model is reasonable for describing your trip.

You Try It - Constant Rate of Change and Extreme Values

Over each 4-hour subinterval of the motion in Part I, the velocity of your car is constant at ± 60 mph. Each subinterval provides an example of a situation where the rate of change or the derivative of a function is constant. Write a brief response to each of the following questions.

1. On the interval where $v = \frac{ds}{dt}$ is positive, what can you say about the value of s?

2. What can you say about the value of s on the interval where $v = \frac{ds}{dt}$ is negative?

3. When $v = \frac{ds}{dt}$ is constant on an interval, what can you say about the relationship between $v = \frac{ds}{dt}$ (the slope of the tangents in the interval) and $\frac{\Delta s}{\Delta t}$ (the slope of any secant line taken inside the interval)?

4. At a hours, the function s reaches its maximum value of 240 miles. What happens to the derivative, $v = \frac{ds}{dt}$, at this point? What are the values of the two one-sided derivatives of s at t = 4? What are the values of $v = \frac{ds}{dt}$ slightly to the left of t = 4 and slightly to the right of t = 4?

5. Over the 8-hour interval, what are the largest and smallest values of s and at what times to they occur?

6. What is the value of $v = \frac{d^2 s}{dt^2}$, the acceleration, during each of the 4-hour subintervals? In the mathematical model, what happens to the acceleration at t = 4 hours? Describe what the acceleration of your car would actually be like when you turn around to come back.

Part II: Constant Acceleration

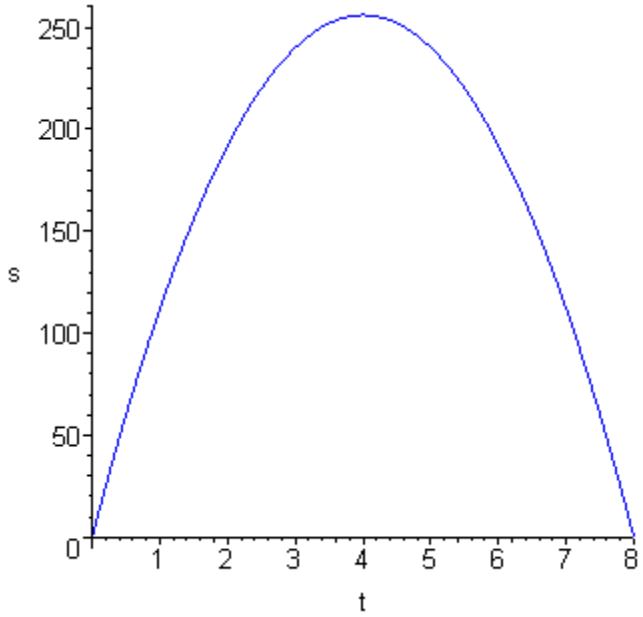
Chapter 3, Section 3

If you throw an object straight up from the ground with an initial speed of 128 ft/s, its height above the ground is given by the following function of time.

NOTE: The commands in this worksheet generate a lot of graphics, which may fill up your computer's memory. Before proceeding with this Part, you should pull down the Edit menu at the top of the screen and select Remove Output (From Worksheet). Then you will have to return to the place in the worksheet where you left off.

```
> s := 128*t - 16*t^2;
  plot(s, t=0..8, labels=["t","s"], color=blue);
```

$$s := 128\,t - 16\,t^2$$



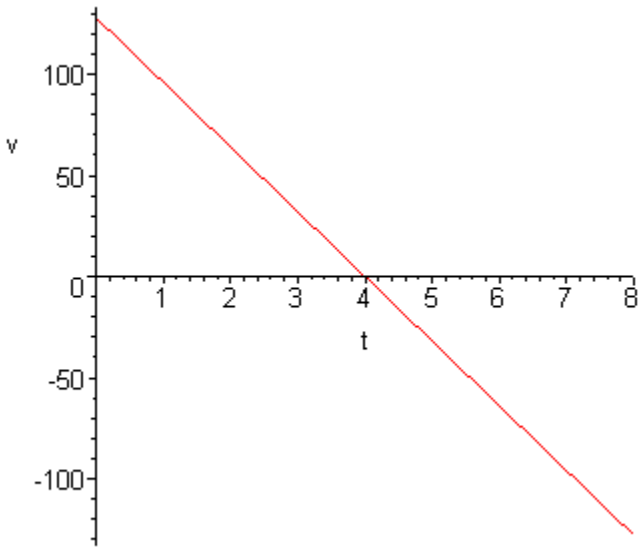
Note that the origin is taken at the ground surface and the positive direction is upward.

Now determine the velocity function and graph it.

> **v:=diff(s,t);**

$$v := 128 - 32\,t$$

> **plot(v, t=0..8, labels=["t","v"], color=red);**



Let's use **velocity( )** to depict the motion. Note that the value for the last argument is 0 since the

motion is not periodic, but if you change the value to 1, the animation will simulate the bouncing motion of a perfectly elastic ball.

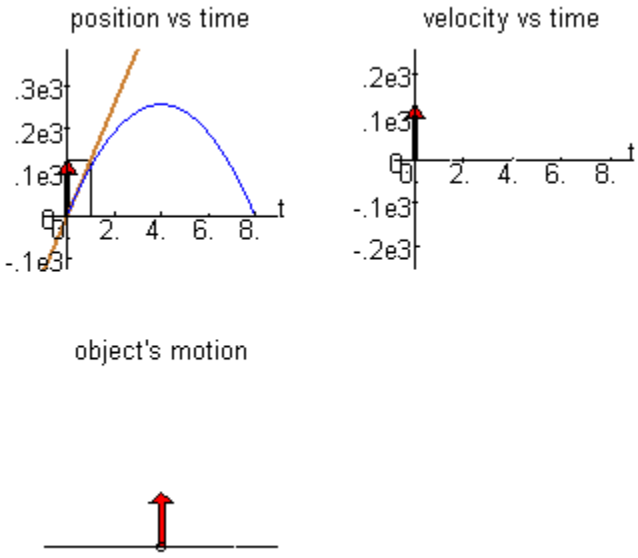
To animate the sequence of graphs:

1) Click on the graphic box that is produced by the next command and the animation toolbar will appear at the top of the screen.
2) Click "Play" button (**large single arrow**) to play the animation.

3) Click on the left or right **double arrow** buttons to speed up or slow down the animation.
4) Click on the **solid square** stop button to stop the animation.
5) You can also view the animation frame by frame by clicking the appropriate **right arrow-bar** button.

The next command may take a little while to execute, depending on the memory size and speed of your computer.

> **velocity(s, t=0..8, 0);**



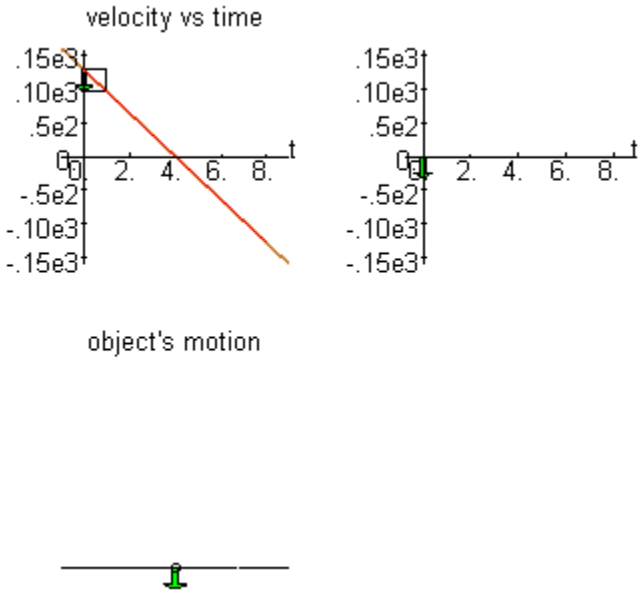
In this example, the rate of change or the slope of the velocity function (i.e., the acceleration) is constant at a = -g = -32 $\frac{ft}{s^2}$.

Acceleration is also a vector quantity. If the acceleration is in the negative direction, the velocity (which can be positive or negative) is decreasing with time. And, if the acceleration is in the positive direction, the velocity (which can be positive or negative) is increasing with time. If the velocity vector and the acceleration vector are in the same direction, then the moving object is speeding up, and if they are in opposite directions, then the object is slowing down.

The **acceleration( )** command produces a sequence of graphs that show: 1) the velocity function, v(t); 2) the acceleration function, a(t); and 3) the object as it moves along a straight line. The acceleration vector is shown on all three graphs. The width of the small black rectangle on the v(t) graph is always one so that the length of the arrow is the slope of the gold tangent line.

You can animate the graphs by following the directions found earlier in this worksheet.

> **acceleration(s, t=0..8, 0);**

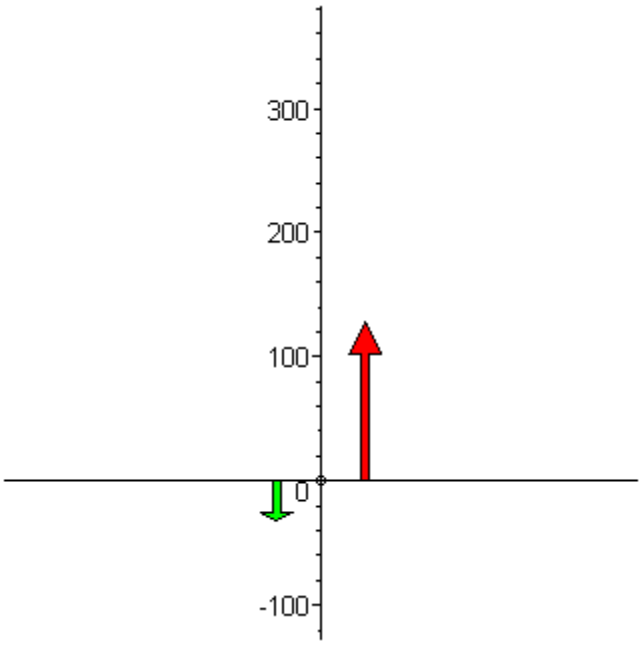


The **posvelacc( )** animation shows the motion of the object, together with its (blue) position, (red) velocity, and (green) acceleration vectors.

You can animate the graphs by following the directions found earlier in this worksheet.

> **posvelacc(s, t=0..8, 0);**

s(blue), v(red) and a(green) vectors



Note that on the way up, the velocity and acceleration are in opposite directions (the velocity is positive and the acceleration is negative) and the object is slowing down, whereas, on the way down the velocity and acceleration are in the same direction (both negative) and the object is speeding up.

Before we continue, there is one last thing we need to point out pertaining to the physics and mathematics of motion. While the purpose of this module is to illustrate the derivative relations between position and velocity and between velocity and acceleration, real world applications usually work the other way around. That is, instead of starting with the position function and differentiating it to find the velocity and acceleration, we usually start with the acceleration function and use it to construct the velocity and position functions. The reason for this is that Isaac Newton taught us, among other things, that the best way to study real world phenomena is to look at the forces that bring about change. In the context of motion problems, this philosophy is embodied in Newton's three laws of motion. The second of Newton's three laws states that if the mass of an object is constant then the total of all the forces acting on the object is equal to its mass times its acceleration, that is, F = m a. This gives us a way to determine the acceleration function for an object by knowing the forces that act on it. Once we know the acceleration we use it to construct the velocity function, and then we us the velocity to construct the position function. The problem then is this: if we know the derivative of a function, how do we determine what the function is? This problem is dealt with in the second big part of calculus - antidifferentiation and integration. You can explore these ideas in a later module (**Motion Along a Straight Line, Part II: Acceleration -> Velocity -> Position**, which treats some of the same motion problems that are in this module, but the other way around.

You Try It - Linear Rate of Change and Extreme Values

Chapter 3, Section 3
Also see Chapter 4, Section 4

The motion in Part II provides an example of a situation where the derivative of a function, i.e., the velocity, is a linear function. Write a brief response to each of the following questions.

1. On what interval is $v = \frac{ds}{dt}$ positive? On what interval is it negative?

2. On the interval where $v = \frac{ds}{dt}$ is positive, what can you say about the value of s?

3. What can you say about the value of s on the interval where $v = \frac{ds}{dt}$ is negative?

4. At t=4 seconds, the function s reaches its maximum value of 256 feet. What happens to the derivative, $v = \frac{ds}{dt}$, at this point?

5. What does the value of $v = \frac{ds}{dt}$ do as t increases? How is this related to the value of $a = \frac{dv}{dt} = \frac{d^2 s}{dt^2}$ and the concavity of the graph of s over the interval of motion?

6. Over the 8-second interval, what are the largest and smallest values of s and at what times to they occur?

Part III: Oscillations

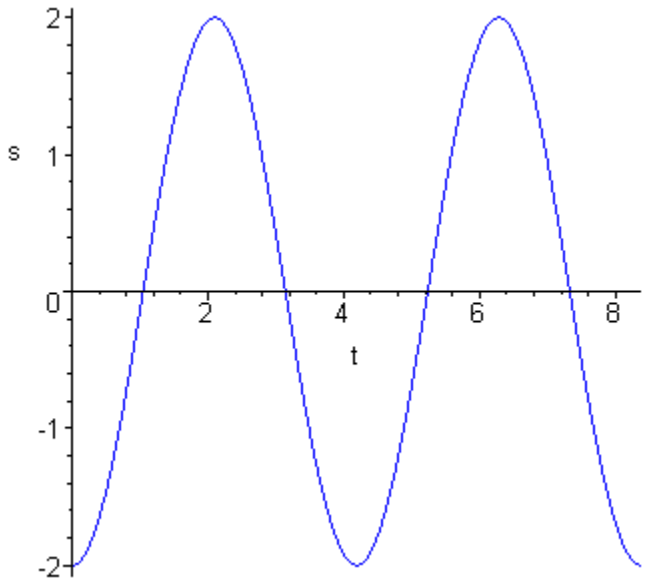
Chapter 3, Section 4, Simple Harmonic Motion

If you hang a mass from a spring, it will eventually come to rest in an equilibrium position, where the weight of the mass is balanced by the tension in the spring. Now pull the mass down slightly and let it go. It will oscillate (move up and down) about the equilibrium position. If there were no air resistance and/or friction in the mass and spring it would oscillate forever with the same amplitude and frequency. Since the motion is periodic, it is not surprising that the periodic trig functions can be used to describe such a motion. For example, if the amplitude of the vibration is 2 centimeters and the period of the oscillations is $\frac{4\pi}{3}$ seconds, then the motion is described by the following function.

NOTE: The commands in this worksheet generate a lot of graphics, which may fill up your computer's memory. Before proceeding with this Part, you should pull down the Edit menu at the top of the screen and select Remove Output (From Worksheet). Then you will have to return to the place in the worksheet where you left off.

> **s:=-2*cos(3*t/2);**
plot(s, t=0..8*Pi/3, labels=["t","s"], color=blue);

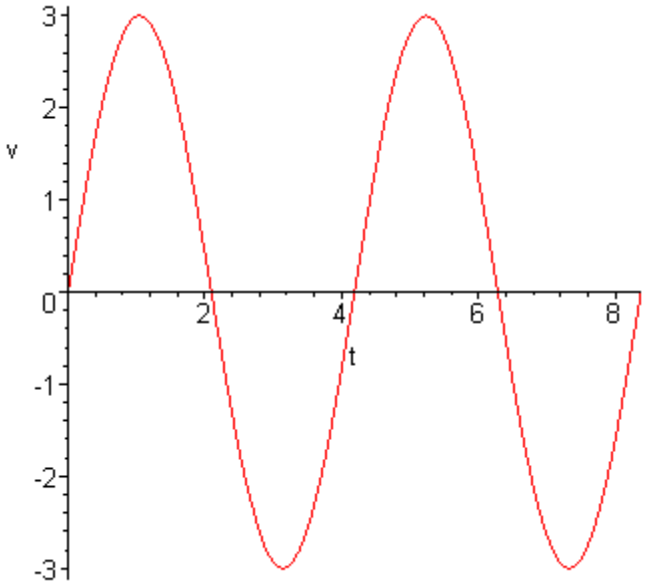
$$s := -2\cos\left(\frac{3t}{2}\right)$$



Now we can calculate the velocity of the mass as it moves up and down and graph it.

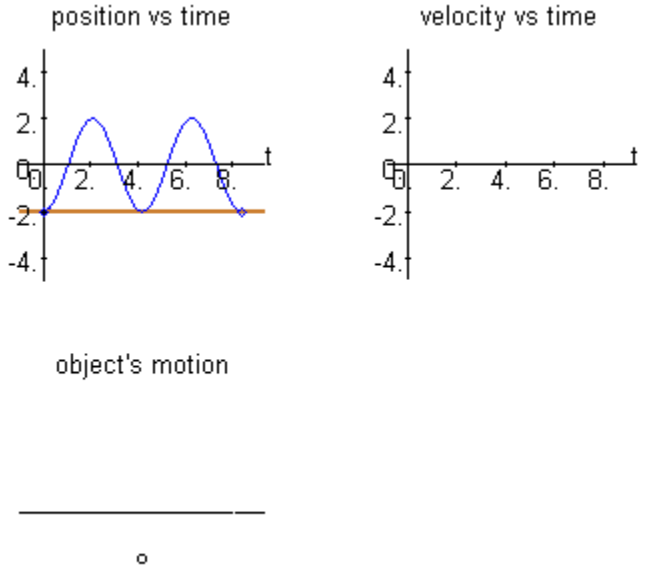
> **v := diff(s, t);**
plot(v, t=0..8*Pi/3, labels=["t","v"], color=red);

$$v := 3\sin\left(\frac{3t}{2}\right)$$



The **velocity( )** animation can be used to depict the motion. Since the motion is periodic, we set the last argument of **velocity( )** to 1 instead of 0. This gives a smooth motion in the animations without a hesitation at the wrap-around, end/beginning point of the time interval.

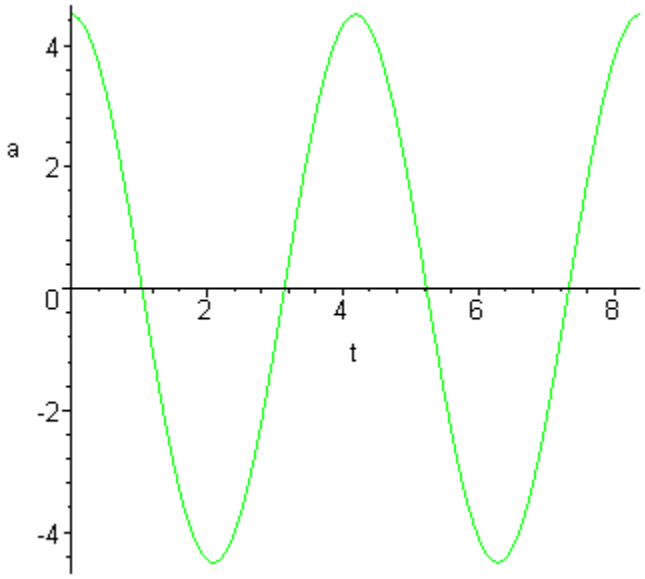
> **velocity(s,t=0..8*Pi/3,1);**



The acceleration of the mass is given by the derivative of the velocity.

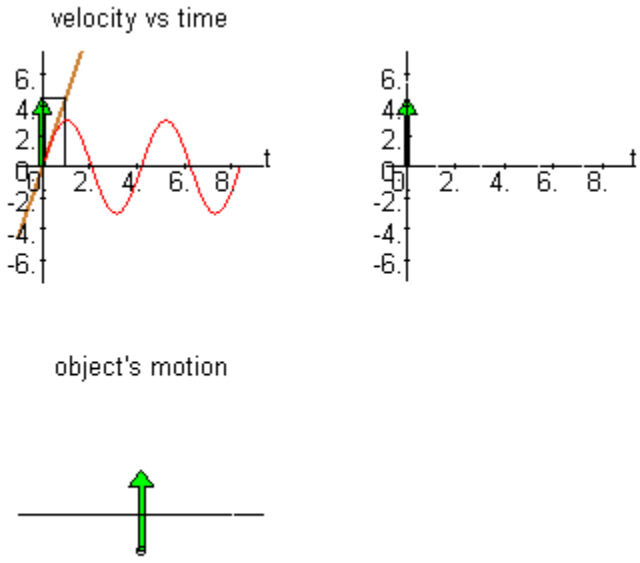
> **a:=diff(v,t);**
plot(a, t=0..8*Pi/3, labels=["t","a"], color=green);

$$a := \frac{9}{2}\cos\left(\frac{3t}{2}\right)$$



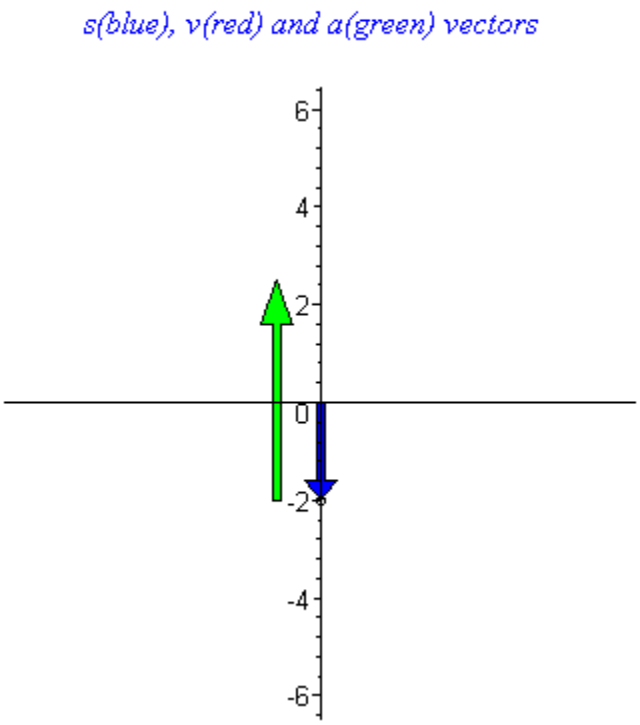
The **acceleration( )** animation illustrates the relationship between the velocity and acceleration as the mass oscillates.

> **acceleration(s,t=0..8*Pi/3,1);**



The **posvelacc( )** animation shows the motion of the object, its position vector, its velocity, and its acceleration.

> **posvelacc(s,t=0..8*Pi/3,1);**

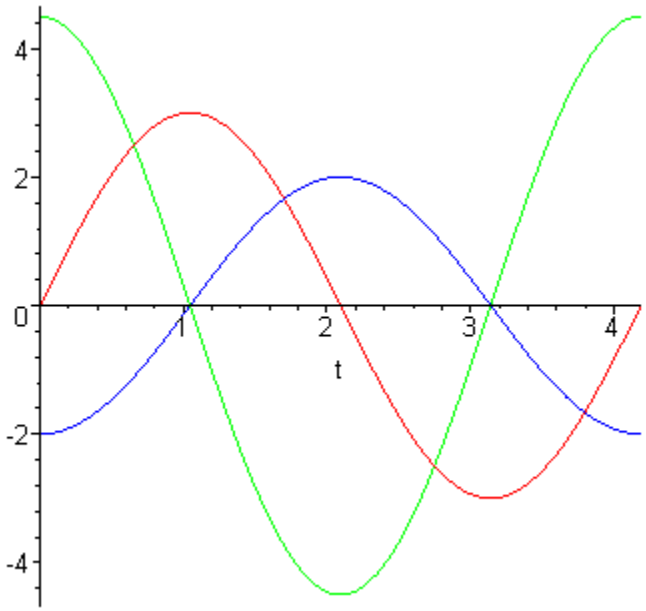


There are some of observations to make about this motion. The first is that the acceleration is in the direction opposite the position. When the mass is below the equilibrium position, the stretch in the spring increases thus exerting a net upward force on the mass, accelerating in that direction. And, when the mass is above the equilibrium position, the stretch in the spring is reduced resulting in a net downward force on the mass, accelerating it in that direction. The second observation is that the acceleration is proportional to the position. The first two observations can be demonstrated mathematically by noting that if $s(t) = -A\cos(\omega t)$ then $a(t) = \frac{d^2 s}{dt^2}$ =

$A\omega^2\cos(\omega t) = \omega^2 \quad A\cos(\omega t) = -\omega^2 s(t)$.

Another thing to note is that the velocity and acceleration are 90 degrees or $\frac{\pi}{2}$ radians out of phase with one another. During the first quarter-cycle of the motion, the velocity and acceleration are both positive and the mass is speeding up, moving upward. During the second-quarter cycle, the velocity is positive and the acceleration is negative, and the mass is still moving upward but now it is slowing down. During the third quarter-cycle, the velocity and acceleration are both negative, and the mass is speeding up moving downward. During the fourth quarter-cycle, the velocity is negative and the acceleration is positive, and the mass is slowing down as it moves downward. Overlaying the graphs of the velocity and acceleration functions shows these relationships.

```
> plot({s,v,a}, t=0..4*Pi/3,color=[blue,red,green]);
```



The velocity is $v(t) = -A\,\omega\,\sin(\omega\,t)$ and the acceleration is $a(t) = -A\,\omega^2\,\cos(\omega\,t)$. From basic trigonometry we know that the sine and cosine functions are 90 degrees or $\frac{\pi}{2}$ radians out of phase with one another.

The final observation is that at the instant when the mass passes the equilibrium position, the velocity reaches its extreme values (positive on the way up and negative on the way down), its speed is at its maximum value, and its acceleration is zero.

You Try It - The Second Derivative, Concavity, and Inflection Points

Chapter 3, Section 3
Also see Chapter 4 Section 4

After you have viewed the **velocity( )**, **acceleration( )**, and **posvelacc( )** animations, write a brief response to each of the following questions.

1. During intervals where the tangent line is rotating clockwise the graph of s is said to be "concave down". Over what intervals is the graph of s concave down? What is the value of

$v = \frac{ds}{dt}$ doing in these intervals? What is the sign of $a = \frac{d^2 s}{dt^2}$ in these intervals?

2. During the intervals where the tangent line is rotating counterclockwise, the graph of s is said to be "concave up". Over what intervals is the graph of s concave up? What is the value of $v = \frac{ds}{dt}$ doing in these intervals? What is the sign of $a = \frac{d^2 s}{dt^2}$ in these intervals?

3. An inflection point is a point on the graph of s where the concavity changes from up to down, or visa versa. Describe the motion of the green tangent line as it passes over the inflection points on the graph of s. What happens to the value of $v = \frac{ds}{dt}$ as the point on the graph passes over the inflection points? What happens to the value of a = $\frac{dv}{dt} = \frac{d^2 s}{dt^2}$ as the point on the graph passes over the inflection points?

4. At what points on the graph of s is the slope the steepest? At what points on the graph of s does the value of $v = \frac{ds}{dt}$ reach its extreme (maximum and minimum) values?

5. Describe the motion of an object that moves up and down along a straight line for each of the following conditions: a) v > 0 and a > 0; b) v > 0 and a < 0; c) v < 0 and a < 0; and d) v < 0 and a > 0. With up taken as the positive direction, specify the direction the object is moving and whether it is speeding up or slowing down.

Part IV: Decaying Oscillations

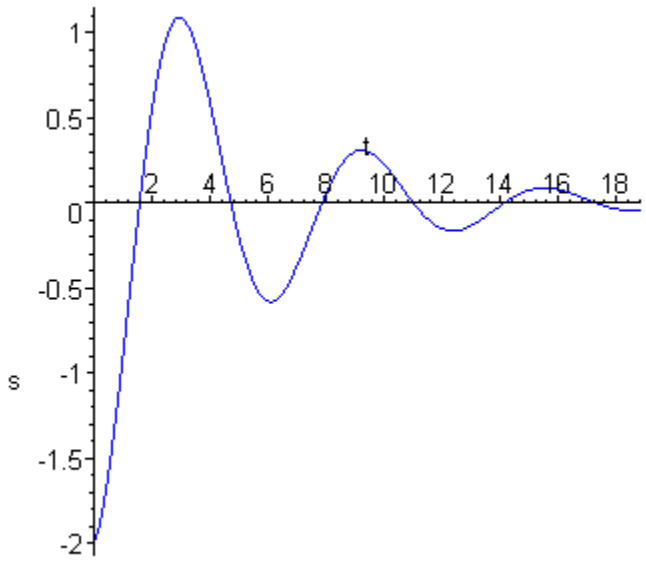
Chapter 3, Section 4

If you wish, you can quit here. The next two Parts, IV and V, are just for fun.

A more realistic model for oscillations of real objects is to account for the loss of energy, which results in a decay in the amplitude of the oscillations. In mechanical systems, this energy loss is usually due to friction (objects rubbing against one another in some way). For the motion of the mass hanging from a spring, a more realistic position function has an amplitude that decays exponentially.

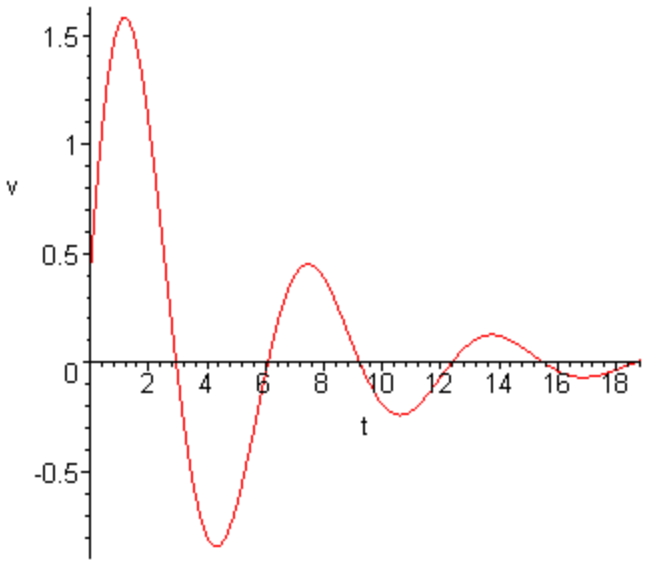
NOTE: The commands in this worksheet generate a lot of graphics, which may fill up your computer's memory. Before proceeding with this Part, you should pull down the Edit menu at the top of the screen and select Remove Output (From Worksheet). Then you will have to return to the place in the worksheet where you left off.

> **s:= -2*exp(-t/5)*cos(t):**
plot(s, t=0..6*Pi, labels=["t","s"], color=blue);



> **v:=diff(s,t);**
plot(v, t=0..6*Pi, labels=["t","v"], color=red);

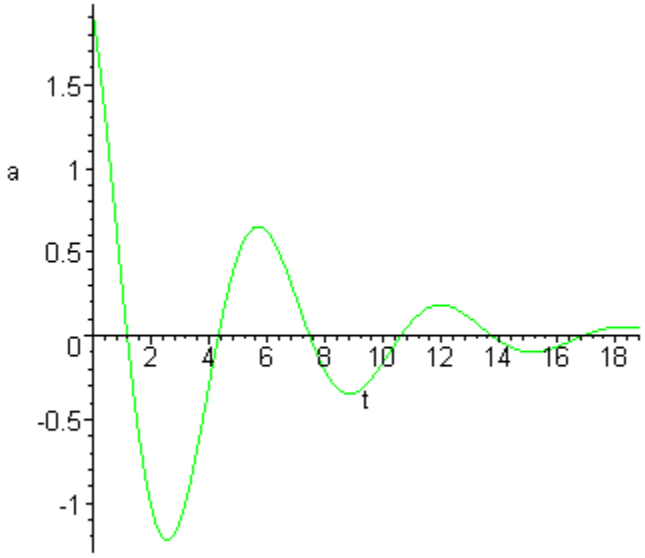
$$v := \frac{2}{5}\,\mathbf{e}^{\left(-\frac{t}{5}\right)}\cos(t) + 2\,\mathbf{e}^{\left(-\frac{t}{5}\right)}\sin(t)$$



> **a:=diff(v,t);**

plot(a, t=0..6*Pi, labels=["t","a"], color=green);

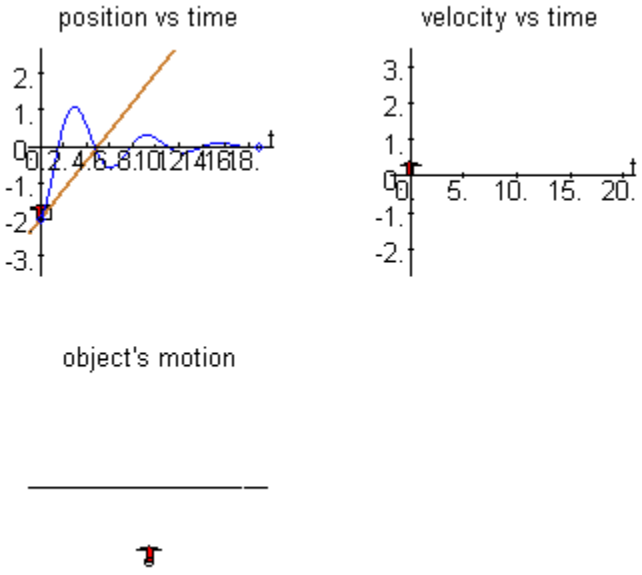
$$a := \frac{48}{25} e^{\left(-\frac{t}{5}\right)} \cos(t) - \frac{4}{5} e^{\left(-\frac{t}{5}\right)} \sin(t)$$



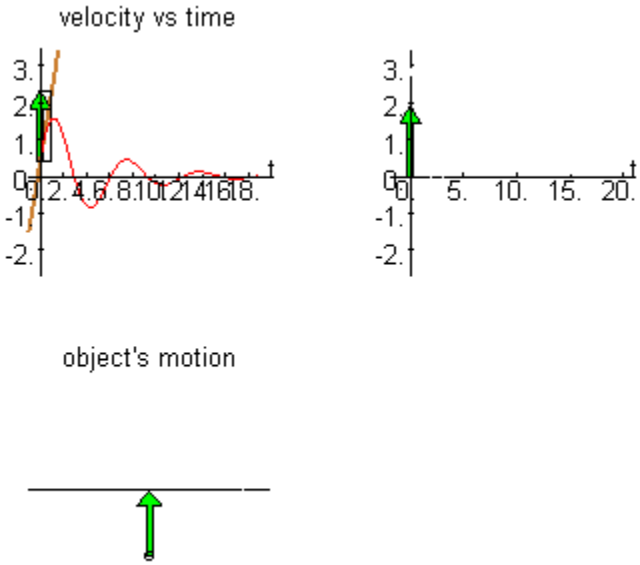
It is apparent that the amplitudes of the velocity and acceleration also decay as time progress.

The motion can be analyzed using the **velocity( ), acceleration( )**, and **posvelacc( )** functions.

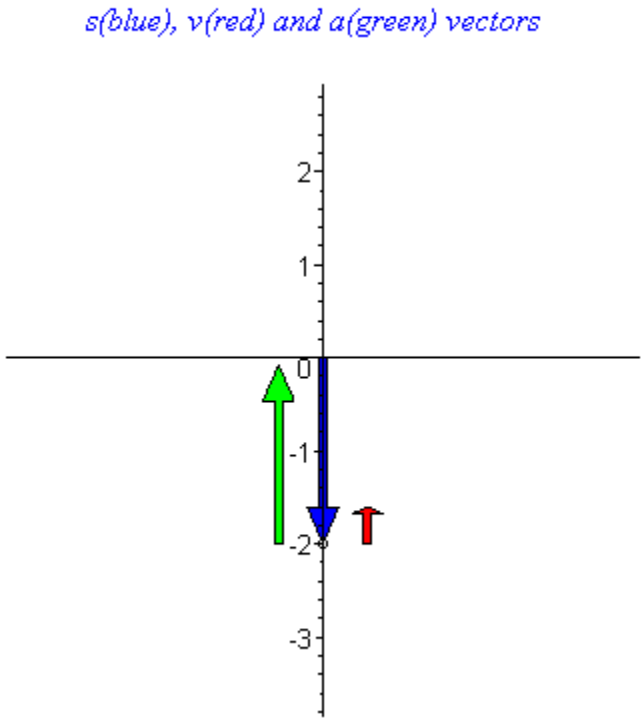
> **velocity(s, t=0..6*Pi, 0);**



> **acceleration(s, t=0..6*Pi, 0);**



> **posvelacc(s, t=0..6*Pi, 0);**



Part V: Earthquake

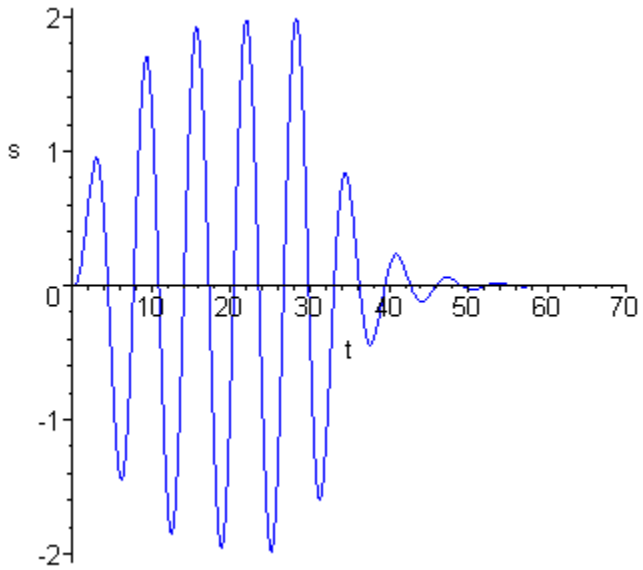
Chapter 3, Section 4

During an earthquake, the floors of building act like masses, and the columns between the floors act like elastic springs. Consequently, the motion of a single floor can be modeled with a mass that is equal to the mass of the floor, and a spring with a stiffness equal to that provided by the columns above and below the floor. In part, the motion will consist of oscillations along a straight line. Initially, when the earthquake begins, the building is at rest and the ground motion puts energy into the building, resulting in back and forth oscillations of increasing amplitude. If the earthquake is long enough in duration, a steady state condition is reached where the amount of energy that goes into the building is equal to the amount dissipated by friction forces as parts of the building rub against one another. During this phase of the earthquake the building sways with oscillations of constant amplitude. Then, when the earthquake stops the oscillations will decay and the building will eventually come to rest (hopefully with little or no damage).

The following function can be used to describe the position of the floor in a building during an earthquake. To see where we got this function see the note at the end of this Part.

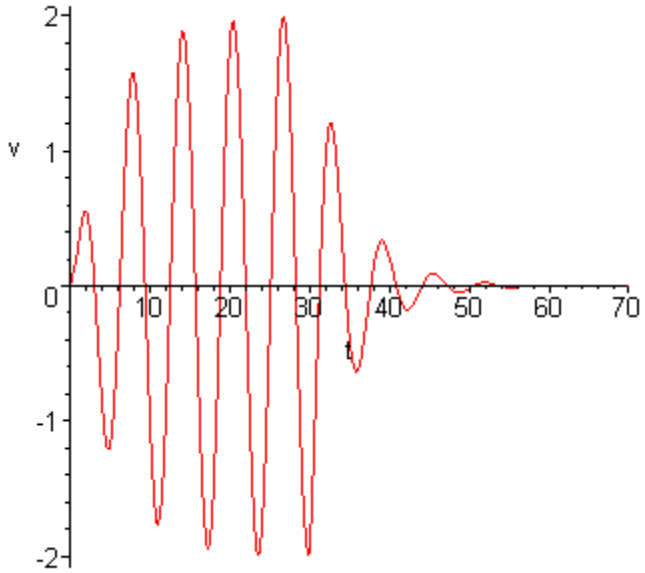
NOTE: The commands in this worksheet generate a lot of graphics, which may fill up your computer's memory. Before proceeding with this Part, you should pull down the Edit menu at the top of the screen and select Remove Output (From Worksheet). Then you will have to return to the place in the worksheet where you left off.

```
> s:=piecewise(t < 30, 2*exp(-t/5)*cos(0.9798*t)-2*cos(t)+0.40825*exp(-t/5)*sin(0.9798*t),t >=
  (-698.8)*exp(-t/5)*cos(0.9798*t) + 478.3*exp(-t/5)*sin(0.9798*t)):
  plot(s, t=0..70, labels =["t", "s"], color=blue);
```

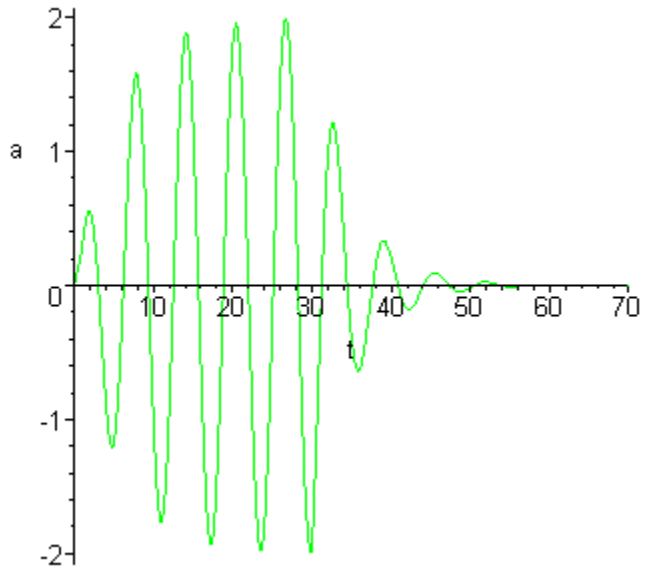


And, the graphs of the velocity and acceleration look like this.

```
> v:=diff(s,t):
  plot(v, t=0..70, labels =["t", "v"], color=red);
```

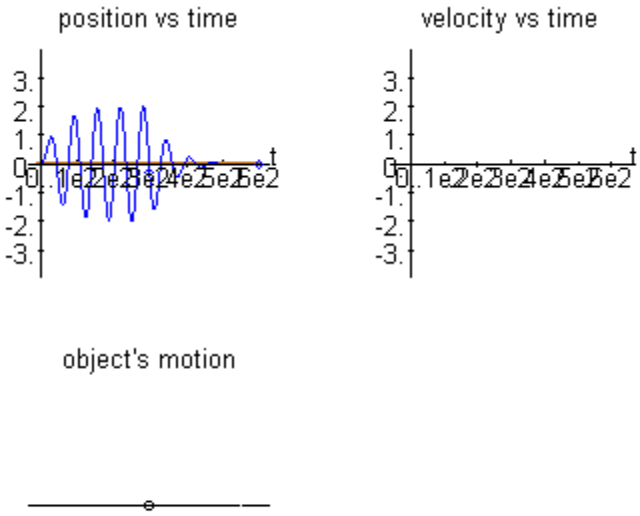


> **a:=diff(s,t):**
plot(a, t=0..70, labels =["t", "a"], color=green);

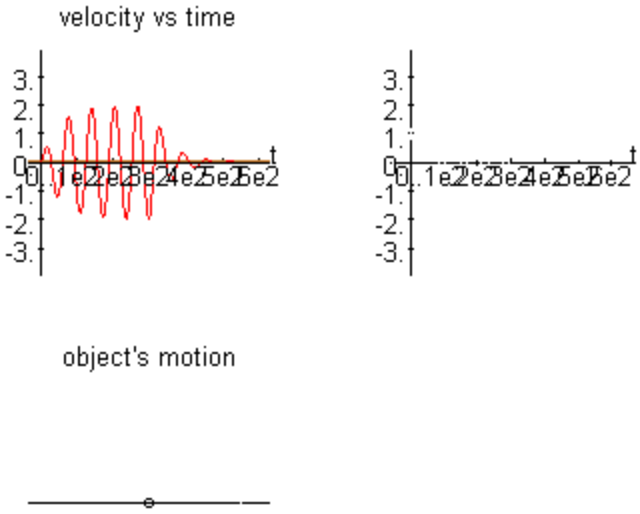


The **velocity( ), acceleration**( ), and **posvelacc( )** functions depict the motion.

> **velocity(s, t=0..60, 0);**

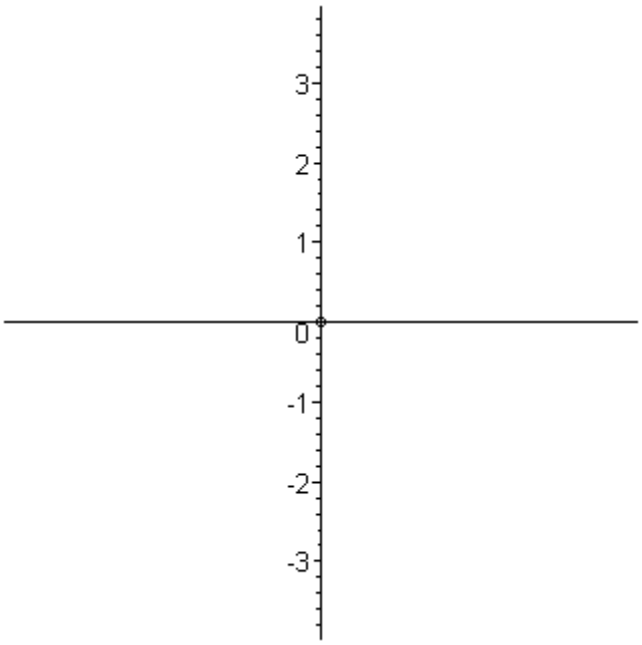


> **acceleration(s, t=0..60, 0);**



> **posvelacc(s, t=0..60, 0);**

s(blue), v(red) and a(green) vectors

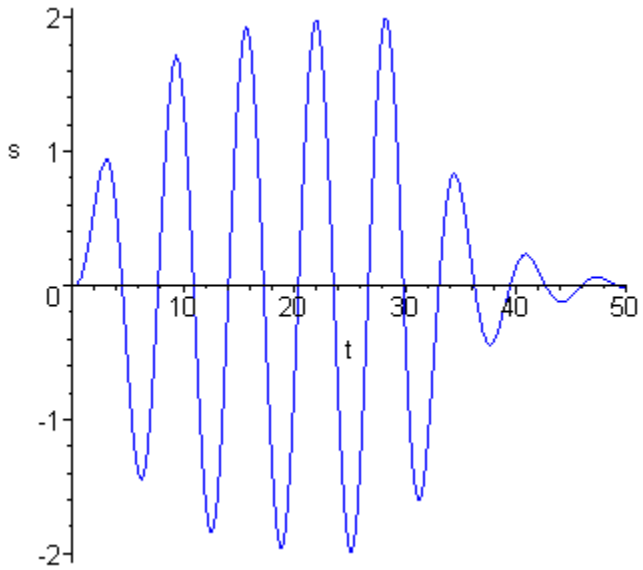


Note: The position function used for the earthquake motion above was obtained by solving the following initial value problem. The ground motion excitation occurs between t = 0 and t = 30, and after t = 30 the motion is free or unforced vibration.

```
> unassign('s'):
  ode1:=(diff(s(t),t,t) + .4*diff(s(t),t) + s(t) = piecewise(t<=30,.8*sin(t),  0 )):
  ic1:=D(s)(0)=0:
  ic2:=s(0)=0:
  dsolve({ode1, ic1, ic2}, s(t));
```

$$s(t) = \begin{cases} \frac{1}{6}\mathbf{e}^{\left(-\frac{t}{5}\right)}\sin\left(\frac{2\sqrt{6}\,t}{5}\right)\sqrt{6} + 2\,\mathbf{e}^{\left(-\frac{t}{5}\right)}\cos\left(\frac{2\sqrt{6}\,t}{5}\right) - 2\cos(t), & t < 30 \\ \frac{1}{6}\mathbf{e}^{\left(-\frac{t}{5}\right)}\sin\left(\frac{2\sqrt{6}\,t}{5}\right)\sqrt{6} + 2\,\mathbf{e}^{\left(-\frac{t}{5}\right)}\cos\left(\frac{2\sqrt{6}\,t}{5}\right) - \mathbf{e}^{\left(-\frac{t}{5}+6\right)}\cos\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} - 30\right) \\ + \frac{5}{12}\mathbf{e}^{\left(-\frac{t}{5}+6\right)}\sqrt{6}\cos\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} - 30\right) - \frac{5}{12}\mathbf{e}^{\left(-\frac{t}{5}+6\right)}\sqrt{6}\cos\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} + 30\right) \\ - \mathbf{e}^{\left(-\frac{t}{5}+6\right)}\cos\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} + 30\right) - \frac{1}{12}\sqrt{6}\sin\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} - 30\right)\mathbf{e}^{\left(-\frac{t}{5}+6\right)} \\ - \frac{1}{12}\sqrt{6}\sin\left(\frac{2\sqrt{6}\,t}{5} - 12\sqrt{6} + 30\right)\mathbf{e}^{\left(-\frac{t}{5}+6\right)}, & 30 \le t \end{cases}$$

```
> plot(rhs(%), t=0..50, labels=["t","s"], color=blue);
```



Part VI: Some Help with Your Homework

Chapter 3, Section 3, Exercises 3

The special commands in this section may be helpful with some of the homework Exercises from Chapter 3, Section 3. For example, we use them to visualize some of the motion for Exercise 3.

NOTE: The commands in this worksheet generate a lot of graphics, which may fill up your computer's memory. Before proceeding with this Part, you should pull down the Edit menu at the top of the screen and select Remove Output (From Worksheet). Then you will have to return to the place in the worksheet where you left off.

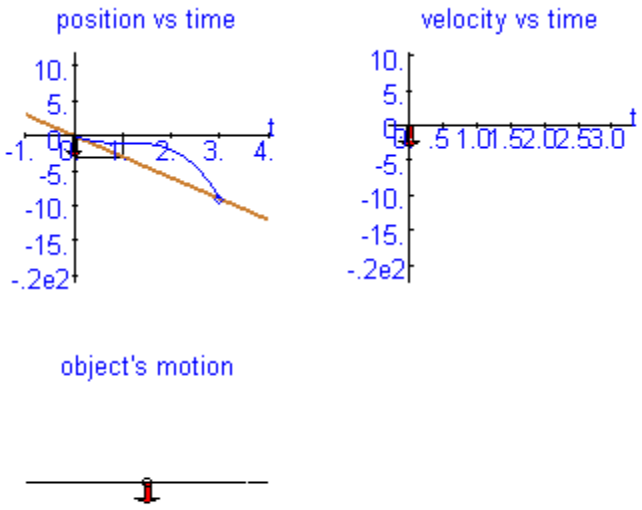
> **s:=-t^3+3*t^2-3*t;**
v:=diff(s,t);
a:=diff(v,t);

$$s := -t^3 + 3t^2 - 3t$$

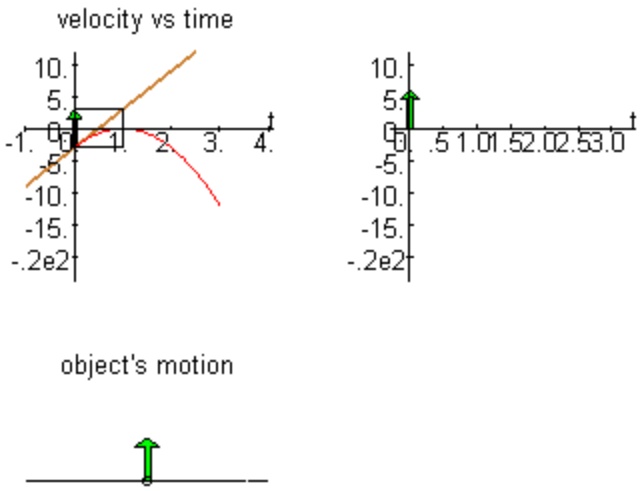
$$v := -3t^2 + 6t - 3$$

$$a := -6t + 6$$

> **velocity(s, t=0..3, 0);**

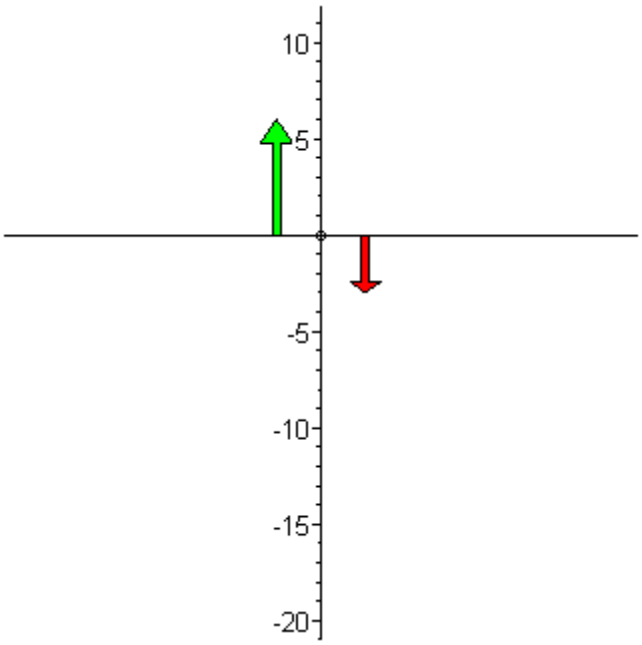


> **acceleration(s, t=0..3, 0);**



> **posvelacc(s, t=0..3, 0);**

s(blue), v(red) and a(green) vectors


10
5
-5
-10
-15
-20

>