Radar Tracking of a Moving Object

Note: You may notice differences between this Maple worksheet and the equivalent Mathematica notebook. These differences were introduced to preserve the content of these modules and were necessary because of major functional differences between Maple and Mathematica.

Introduction

OBJECTIVE: Explore how to compute velocity and acceleration of a moving object whose position is given in polar coordinates. Animate the trajectory of a moving object together with its position, velocity, and acceleration vectors.

In many practical applications, including the radar detection of a moving object, the position of an object is specified in polar coordinates with an angle and a distance. In this module, you will learn how to convert the position specification from polar coordinates to Cartesian coordinates and how to calculate the velocity and acceleration vectors. Animations are used to study several different motions, showing the object's position, velocity, and acceleration vectors as time progresses.

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, and then shut down *Maple* and start it up again.

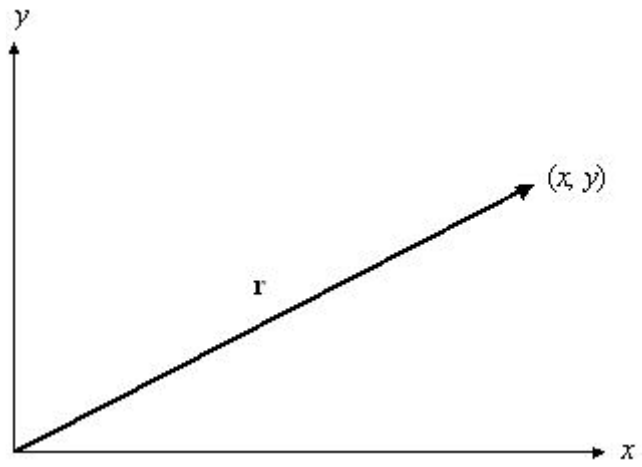
Part I: Converting from Polar to Cartesian Coordinates

Cartesian Coordinates

The location of an object in a two-dimensional plane can be specified with two coordinates, x and y. These are the familiar Cartesian coordinates of a point in a plane. When the Cartesian coordinate values are placed in an array like this $[x, y]$, the array is called a position vector, and it is denoted by the symbol **r**. That is, $\mathbf{r} = [x, y]$.

The order of the entries in a vector is important. The position [1, 2] is different from the position [2, 1]. Note that **r**, the symbol used to represent the vector, is a bold, lowercase letter. In contrast, x and y, the symbols used to represent the single numbers in the vector, are in italics and are not bold. The numbers x and y are called the *components* of the vector. An alternative notation for a position vector, typically used in physics and engineering, is $\mathbf{r} = x\,\mathbf{i} + y\,\mathbf{j}$. This representation of the position vector contains exactly the same information as the $[x, y]$ notation, but represents it in a different way.

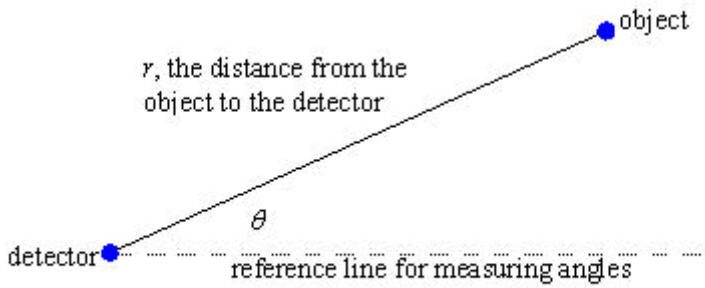
Geometrically, a position vector, **r**, is shown as an arrow in the Cartesian plane, extending from the origin to the position with coordinates x and y, as shown in the next figure.



For further exploration of moving objects in Cartesian coordinates, refer to "Projectile Motion," a JAVA applet included in this supplement.

Polar Coordinates

Another natural way to specify the position of an object in a plane is with polar coordinates. One application of polar coordinates is in the use of radar (radio-detecting-and-ranging) to track a moving object. Radar tracks an object by bouncing a radio-frequency light wave off of the object to measure its distance from the radar detector and its angular position relative to some fixed reference line through the detector.



The angle, θ, is positive if it is counterclockwise from the fixed reference line and negative if it is clockwise.

Using polar coordinates, we specify positions or points in a plane by a distance r from some fixed reference point, called the pole, and an angle θ from a fixed reference line through the *pole*. The values of r and θ are called the *polar coordinates* of the position or point.

In the strict sense, the r-coordinate of a point in polar coordinates can be positive, negative, or zero. (See Section 9.5, "Polar Coordinates and Graphs," in the textbook.) However, in this module we make the assumption that r measures distance from the pole, therefore, we restrict r to be greater than or equal to 0.

Converting from Polar to Cartesian Coordinates

Given the location of an object specified in polar coordinates, we would like to find its Cartesian coordinates. Therefore, we write a pair of transformation equations that will convert a pair of polar coordinates (r, θ) into Cartesian coordinates (x, y). The positive x-axis in the Cartesian reference frame is taken to be along the polar reference line with the origin at the pole, and the positive y-axis is in the polar direction, $\theta = \frac{\pi}{2}$. In addition, we treat the variables r and θ as parametric functions of time. First we take care of some housekeeping by loading packages into memory and restricting r to be nonnegative.

```
> restart;
  with(plots):
  with(plottools):
  interface(showassumed=0):
  assume(r(t)>=0);
```

Warning, the name changecoords has been redefined

Warning, the assigned name arrow now has a global binding

If $r = r(t)$ and $\theta = \theta(t)$, then the transformations to Cartesian coordinates are:

```
> x:=t->r(t)*cos(theta(t));
  y:=t->r(t)*sin(theta(t));
```

$$x := t \to r(t)\cos(\theta(t))$$

$$y := t \to r(t)\sin(\theta(t))$$

Let's consider an example. First, we specify the position of an object in polar coordinates.

```
> r(t):=t:
  theta:=t->t^2:
  rvectorpolar:=[r(t), theta(t)];
```

$$rvectorpolar := [t, t^2]$$

And we convert the position to a vector in Cartesian coordinates using the transformation equations that we defined above.

```
> rvectorCartesian:=[x(t), y(t)];
```

$$rvectorCartesian := [t\cos(t^2), t\sin(t^2)]$$

You Try It: Part I

Given the following time functions for the polar coordinates of a moving object, form expressions for the vector position of the object in polar and Cartesian coordinates.

```
> unassign('r','theta','rvectorpolar','rvectorCartesian');
  r(t):=sin(t^2);
  theta(t):=2*t;
```

$$r(t) := \sin(t^2)$$

$$\theta(t) := 2t$$

Part II: Position, Velocity, Speed, and Direction

Now that we have a vector expression for the position vector in terms of $r(t)$ and $\theta(t)$, we can determine the velocity, speed, and direction of travel for the moving object. Here we express the position vector in Cartesian coordinates.

```
> unassign('r','theta','x','y');
  x:=r(t)*cos(theta(t));
  y:=r(t)*sin(theta(t));
  rvectorCartesian:=[x, y];
```

$$x := r(t)\cos(\theta(t))$$

$$y := r(t)\sin(\theta(t))$$

$$rvectorCartesian := [r(t)\cos(\theta(t)), r(t)\sin(\theta(t))]$$

If a radar detector is tracking a moving object, then r and θ vary with time. The derivative $\frac{dx}{dt}$ measures the rate of change of the x coordinate of the object. This derivative is called the velocity of the object in the x direction and is denoted by v_x. Similarly, $\frac{dy}{dt}$ is the rate of change of the y coordinate of the object; it is called the velocity of the object in the y direction and is denoted by v_y. Next we determine expressions for v_x and v_y in terms of r and θ and their time derivatives, and we write an expression for the velocity vector.

```
> vx:=diff(x,t);
```

$$vx := \left(\frac{d}{dt}r(t)\right)\cos(\theta(t)) - r(t)\sin(\theta(t))\left(\frac{d}{dt}\theta(t)\right)$$

```
> vy:=diff(y,t);
```

$$vy := \left(\frac{d}{dt}r(t)\right)\sin(\theta(t)) + r(t)\cos(\theta(t))\left(\frac{d}{dt}\theta(t)\right)$$

```
> vvectorCartesian:=[vx,vy];
```

$$vvectorCartesian := \left[\left(\frac{d}{dt}r(t)\right)\cos(\theta(t)) - r(t)\sin(\theta(t))\left(\frac{d}{dt}\theta(t)\right), \left(\frac{d}{dt}r(t)\right)\sin(\theta(t)) + r(t)\cos(\theta(t))\left(\frac{d}{dt}\theta(t)\right)\right]$$

Or, more simply, we can take the time derivative of the position vector, **rvectorCartesian(t).**

```
> vvectorCartesian:=diff(rvectorCartesian,t);
```

$$vvectorCartesian := \left[\left(\frac{d}{dt}r(t)\right)\cos(\theta(t)) - r(t)\sin(\theta(t))\left(\frac{d}{dt}\theta(t)\right), \left(\frac{d}{dt}r(t)\right)\sin(\theta(t)) + r(t)\cos(\theta(t))\left(\frac{d}{dt}\theta(t)\right)\right]$$

The speed of the object is the magnitude of the velocity vector. First, we define a function to calculate the magnitude of a vector, and then we use it to calculate the speed of the moving object.

> **magnitude:=vec->sqrt(vec[1]^2+vec[2]^2);**

$$magnitude := vec \rightarrow \sqrt{vec_1{}^2 + vec_2{}^2}$$

We use **magnitude( )** to find the speed of the object.

> **speed:=simplify(magnitude(vvectorCartesian));**

$$speed := \sqrt{\left(\frac{d}{dt}r(t)\right)^2 + \left(\frac{d}{dt}\theta(t)\right)^2 r(t)^2}$$

The direction of motion of the object is the velocity divided by the speed.

> **vdirection:=vvectorCartesian/speed;**

$$vdirection := \frac{\left[\left(\frac{d}{dt}r(t)\right)\cos(\theta(t)) - r(t)\sin(\theta(t))\left(\frac{d}{dt}\theta(t)\right), \left(\frac{d}{dt}r(t)\right)\sin(\theta(t)) + r(t)\cos(\theta(t))\right(}{\sqrt{\left(\frac{d}{dt}r(t)\right)^2 + \left(\frac{d}{dt}\theta(t)\right)^2 r(t)^2}}$$

The magnitude of the position vector, **rvectorCartesian(t)**, is the distance of the object or point from the pole.

> **distance:=simplify(magnitude(rvectorCartesian),assume(r(t)>=0));**

$$distance := r(t)$$

And the direction of the position vector is the unit vector in the direction of **rvectorCartesian(t)**.

> **rdirection:=expand(rvectorCartesian/distance);**

$$rdirection := [\cos(\theta(t)), \sin(\theta(t))]$$

Part III: A Specific Trajectory

The path of motion of a moving object is called its trajectory. Let's plot the trajectory of an object. First, we define a position function in terms of the parametric functions $r(t)$ and $\theta(t)$ and then graph the trajectory.

> **r:=.5*t;**

$$r := 0.5\,t$$

> **theta:=.5*t^(5/4);**

$$\theta := 0.5\,t^{\left(\frac{5}{4}\right)}$$

> **x:=r*cos(theta);**

$$x := 0.5\,t\cos\left(0.5\,t^{\left(\frac{5}{4}\right)}\right)$$

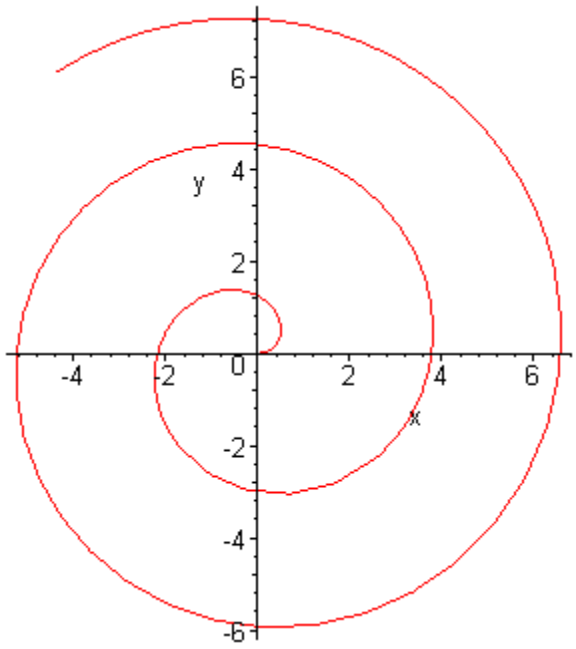
> **y:=r*sin(theta);**

$$y := 0.5\,t\sin\left(0.5\,t^{\left(\frac{5}{4}\right)}\right)$$

> **rvectorCartesian:=[x,y];**

$$rvectorCartesian := \left[0.5\,t\cos\left(0.5\,t^{\left(\frac{5}{4}\right)}\right),\ 0.5\,t\sin\left(0.5\,t^{\left(\frac{5}{4}\right)}\right)\right]$$

> **plot([op(rvectorCartesian), t=0..15], labels=["x","y"], scaling=constrained);**

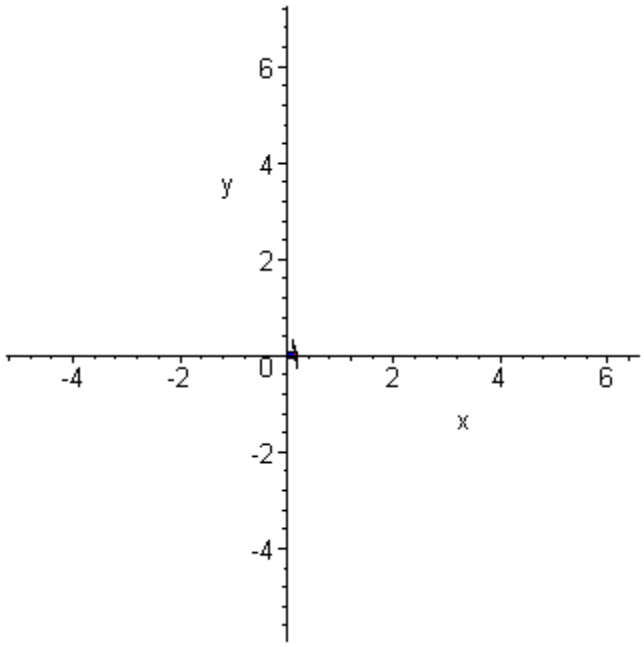


Part IV: Visualizing Position, Velocity, and Acceleration

In *Maple*, you can create several plot objects and plot them together using the **display** command. You can then create a series of multiple plots and animate them by setting **insequence=true.** We create a small loop to generate a series of graphs as time increases by a specified increment. Each graph in the sequence shows the path of motion up to time *t* and the position vector at time *t*.

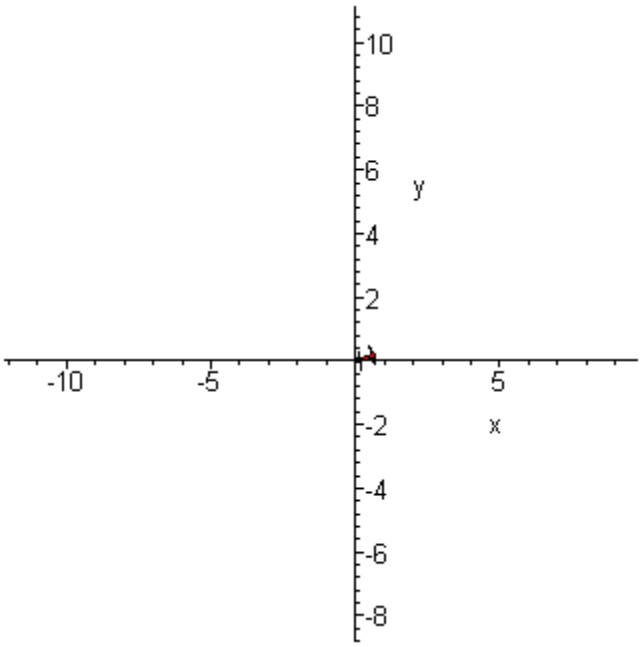
To animate the graph, click anywhere on the plot to display the animation toolbar at the top of the worksheet. From the toolbar, click on the **play button** to view the animation.

```
> window:=12: timel:=12:
  ax:=unapply(x,t):
  ay:=unapply(y,t):
  for i from 1 to 40 do
    p[i]:=plot([op(rvectorCartesian), t=0..i*15/40], labels=["x","y"]):
    a1:=evalf(ax(i*15/40)):
    a2:=evalf(ay(i*15/40)):
    a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
    ani[i]:=display({p[i],a[i]}):
  od:
  display(seq(ani[i], i=1..40), insequence=true);
```



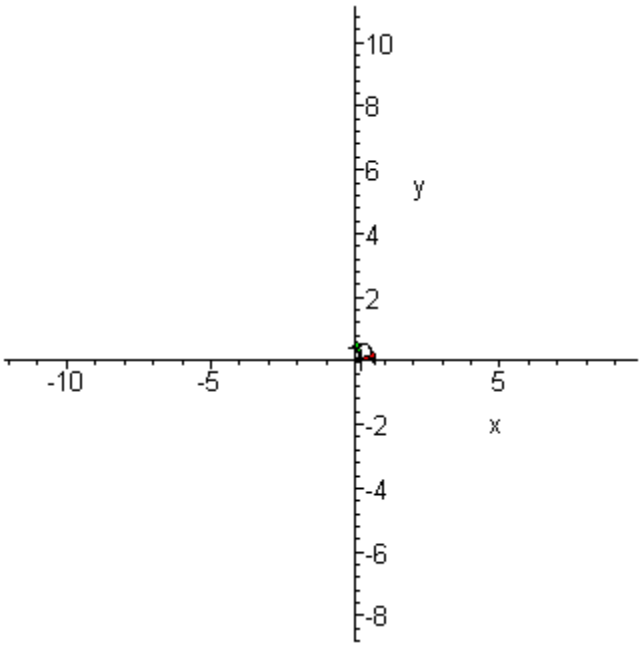
Now let's add the velocity vector to the animation. We want the tail of the velocity vector to be at the head of the position vector; therefore, the arrow should be drawn from **rvectorCartesian(t)** to **rvectorCartesian(t) + vvectorCartesian(t)**.

```
> vvectorCartesian:=diff(rvectorCartesian,t):
  for i from 1 to 40 do
    p[i]:=plot([op(rvectorCartesian), t=0..i*15/40], labels=["x","y"]):
    a1:=evalf(ax(i*15/40)):
    a2:=evalf(ay(i*15/40)):
    a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
    b1:=evalf(subs(t=i*15/40,rvectorCartesian)+subs(t=i*15/40, vvectorCartesian)):      b
  [i]:=arrow([a1,a2], b1, .1,.6,.2, color=red):
    ani[i]:=display({p[i],a[i],b[i]}):
  od:
  display(seq(ani[i], i=1..40), insequence=true);
```



What the heck, why not add the acceleration?

```
> avectorCartesian:=diff(vvectorCartesian,t):
  for i from 1 to 40 do
    p[i]:=plot([op(rvectorCartesian), t=0..i*15/40], labels=["x","y"]):
    a1:=evalf(ax(i*15/40)):
    a2:=evalf(ay(i*15/40)):
    a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
    b1:=evalf(subs(t=i*15/40,rvectorCartesian)+subs(t=i*15/40, vvectorCartesian)):
    b[i]:=arrow([a1,a2], b1, .1,.6,.2, color=red):
    c1:=evalf(subs(t=i*15/40,rvectorCartesian)+subs(t=i*15/40, avectorCartesian)):
    c[i]:=arrow([a1,a2], c1, .1,.6,.2, color=green):
    ani[i]:=display({p[i],a[i],b[i],c[i]}):
  od:
  display(seq(ani[i], i=1..40), insequence=true);
```



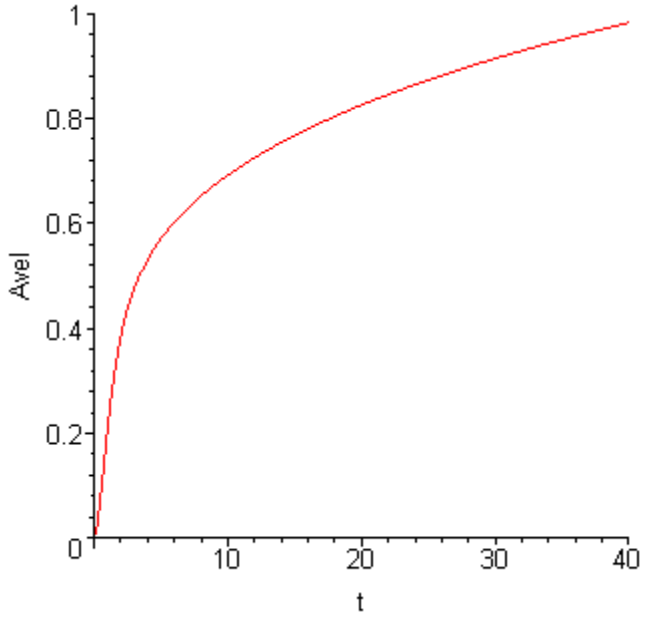
Part V: Speeding Up or Slowing Down? Turning Left or Turning Right?

A few things are worth noting at this point:

1) The velocity is always along the tangent to the path of motion.

2) The acceleration has a nonzero component in the direction of the velocity. We show this by plotting the dot or scalar product of the acceleration vector with the unit vector in the direction of the velocity over the interval of time during the motion. This gives the component of the acceleration in the direction of the velocity, which we call **Avel**.

```
> timel:=40:
  speed:=sqrt(vvectorCartesian[1]^2+vvectorCartesian[2]^2):
  vdirection:=simplify([vvectorCartesian[1]/speed, vvectorCartesian[2]/speed]):
  temp:=linalg[dotprod](avectorCartesian, vdirection):
  plot(temp, t=0..timel, labels=["t","Avel"],labeldirections=[HORIZONTAL,VERTICAL]);
```



You should note that **Avel** is always positive, indicating that the component of the acceleration along the line of the velocity is, in fact, in the same direction as the velocity (instead of opposite it, as would be the case if the dot product were negative). Whenever this situation occurs, the object is speeding up. If the dot product is negative, then the acceleration has a component in the direction opposite that of the velocity, and the object is slowing down.

3) The object turns in the direction of the component of the acceleration that is perpendicular to the velocity. In the preceding example, the object is turning to the left, and the acceleration points to the left. The opposite is true when the object is turning to the right.

You Try It: Part V

To assist you, we copy the commands you need in the following sections. Change the entries below and execute the commands in all of the sections that follow.

Change the polar coordinate functions **r** and θ to generate the following motions.

1. The same motion in Part IV, with the spiral clockwise instead of counterclockwise.

2. The same motion as in Part IV, with the *r*-coordinate starting at 10 and decreasing at the same rate it increased in Part IV.

3. For each motion, use a plot of **Avel** to indicate when the object is speeding up and when it is slowing down.

```
> restart;
  with(plots):
  with(plottools):
```

Warning, the name changecoords has been redefined

Warning, the assigned name arrow now has a global binding

> **r := .5*t;**

$$r := 0.5\,t$$

> **theta:=.5*t^(5/4);**

$$\theta := 0.5\,t^{\left(\frac{5}{4}\right)}$$

> **x:=r*cos(theta);**

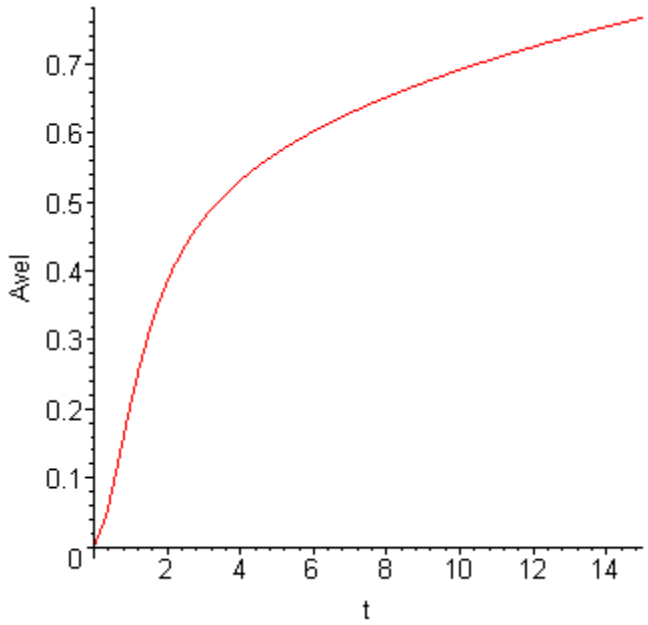
$$x := 0.5\,t\cos\left(0.5\,t^{\left(\frac{5}{4}\right)}\right)$$

> **y:=r*sin(theta);**

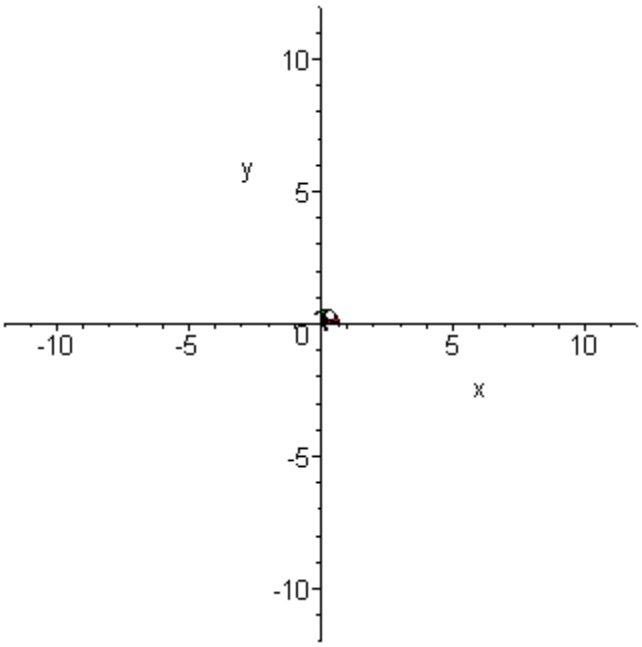
$$y := 0.5\,t\sin\left(0.5\,t^{\left(\frac{5}{4}\right)}\right)$$

> **rvectorCartesian:=[x,y]:**
vvectorCartesian:=diff(rvectorCartesian,t):
avectorCartesian:=diff(vvectorCartesian,t):
speed:=sqrt(vvectorCartesian[1]^2+vvectorCartesian[2]^2):
vdirection:=simplify([vvectorCartesian[1]/speed,vvectorCartesian[2]/speed]):

> **window:=12:**
timel:=15:
plot(linalg[dotprod](avectorCartesian, vdirection), t=0..timel
, labels=["t","Avel"],labeldirections=[HORIZONTAL,VERTICAL]);




```
> ax:=unapply(x,t):
  ay:=unapply(y,t):
  for i from 1 to 40 do
    p[i]:=plot([op(rvectorCartesian), t=0..i*timel/40], labels=["x","y"]):
    a1:=evalf(ax(i*timel/40)):
    a2:=evalf(ay(i*timel/40)):
    a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
    b1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, vvectorCartesian)):
    b[i]:=arrow([a1,a2], b1, .1,.6,.2, color=red):
    c1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, avectorCartesian)):
    c[i]:=arrow([a1,a2], c1, .1,.6,.2, color=green):
    ani[i]:=display({p[i],a[i],b[i],c[i]}):
  od:
  display(seq(ani[i], i=1..40), insequence=true, view=[-window..window, -window..window]);
```



Part VI: Circular Motion

We put all of the commands into the sections that follow so we can try some different curves without having to scroll back up to re-execute commands. First we generate a parametric plot of the trajectory. From this we select values for the window size and time limit for use in the following sections, which generate the graphs for the animation.

The motion we consider as an example is along a circular path; the object moves clockwise, and it is speeding up. First, we plot the trajectory.

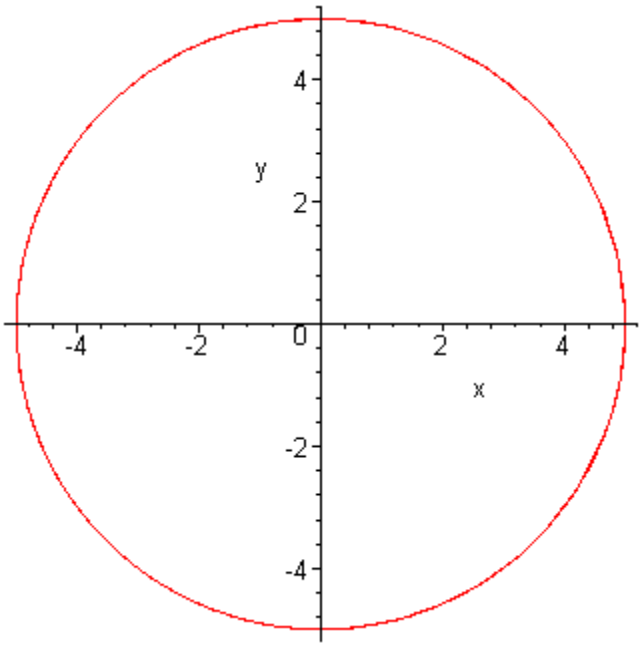
```
> restart;
  with(plots):
  with(plottools):
```

Warning, the name changecoords has been redefined

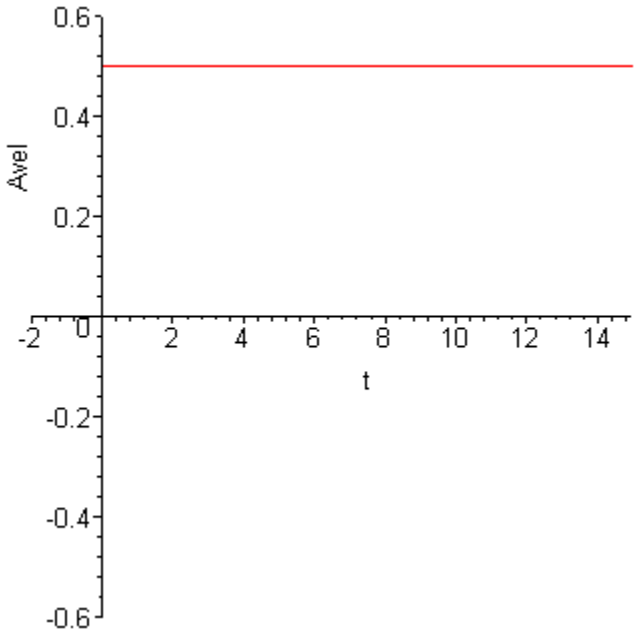
Warning, the assigned name arrow now has a global binding

```
> unassign('r,theta,x,y,rvectorCartesian, vvectorCartesian, avectorCartesian');
  timel:=16*Pi/3:
  theta:=t->-0.05*t^2:
  r:=t->5:
  x:=t->r(t)*cos(theta(t)):
  y:=t->r(t)*sin(theta(t)):
  rvectorCartesian:=[x(t),y(t)];
  plot([op(rvectorCartesian), t=0..timel], labels=[x,y], scaling=constrained);
```

$$rvectorCartesian := [5\cos(0.05\,t^2), -5\sin(0.05\,t^2)]$$



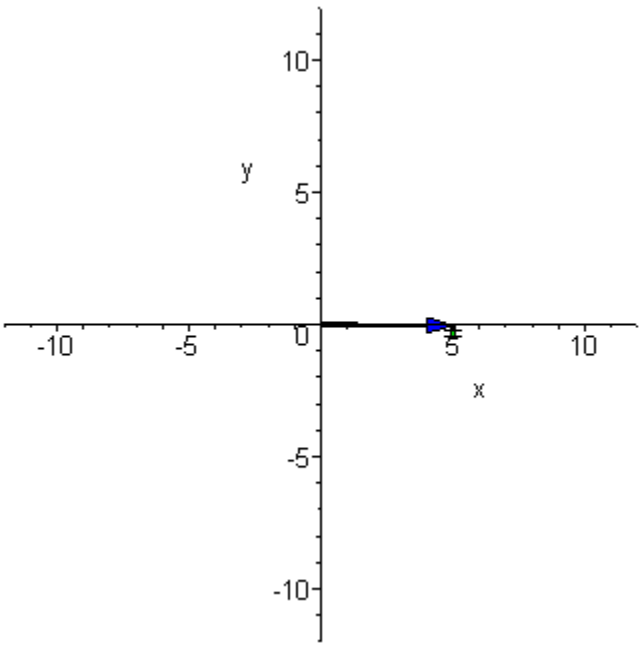
> **window:=12:**
vvectorCartesian:=diff(rvectorCartesian,t):
avectorCartesian:=diff(vvectorCartesian,t):
speed:=simplify(sqrt(vvectorCartesian[1]^2+vvectorCartesian[2]^2)):
vdirection:=[vvectorCartesian[1]/speed,vvectorCartesian[2]/speed]:
prod:=linalg[dotprod](avectorCartesian, vdirection):
plot(prod, t=0..15, labels=["t","Avel"],labeldirections=[HORIZONTAL,VERTICAL],view
2..15,-0.6..0.6]);



The dot product of the acceleration with the direction vector is positive, so the object is speeding up.

Next, we animate the motion showing the position, velocity, and acceleration vectors on the moving point.

```
> for i from 1 to 40 do
  p[i]:=plot([op(rvectorCartesian), t=0..i*timel/40], labels=["x","y"]):
  a1:=x(i*timel/40):
  a2:=y(i*timel/40):
  a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
  b1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, vvectorCartesian)):
  b[i]:=arrow([a1,a2], b1, .1,.6,.2, color=red):
  c1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, avectorCartesian)):
  c[i]:=arrow([a1,a2], c1, .1,.6,.2, color=green):
  ani[i]:=display({p[i],a[i],b[i],c[i]}):
 od:
 display(seq(ani[i], i=1..40), insequence=true, view=[-window..window, -window..window]);
```



You Try It: Part VI

To assist you, we copy the commands you need in the following sections.

Change the polar coordinate functions **r(t)**and θ **(t)** to generate the following motions.

1. A counterclockwise circular path with a radius of 10 and the angle θ **(t)** increasing the same way it did in Part VI.

2. A clockwise circular path with a radius of 10 and the angle θ **(t)** decreasing at the same rate it increased in Part VI.

3. Try this one: $r(t) = 10$ and $\theta(t) = \frac{t(14-t)}{12}$. What do you notice about the component of the acceleration in the direction of the velocity? When is the component of the acceleration that is perpendicular to the velocity the largest? When is it the smallest? Is the velocity vector ever equal to the zero-vector? Is the acceleration vector ever equal to the zero-vector?

4. For each motion, use the plot of **Avel** to indicate whether the object is speeding up or slowing down.

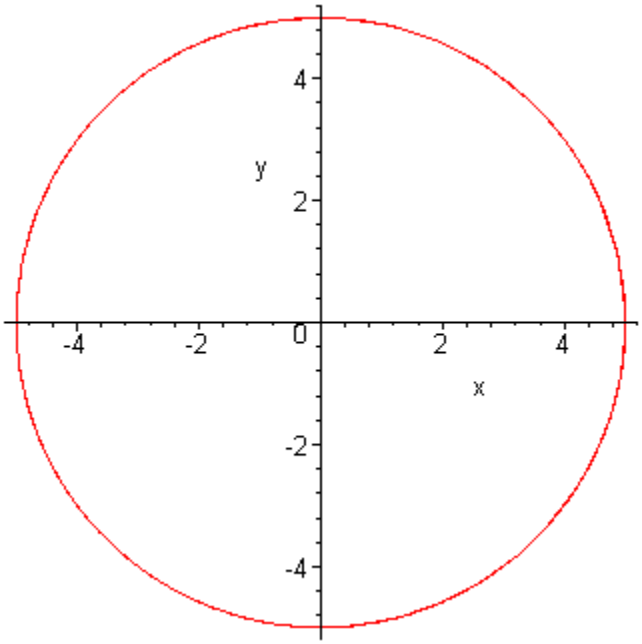
```
> restart;
  with(plots):
  with(plottools):
```

Warning, the name changecoords has been redefined

Warning, the assigned name arrow now has a global binding

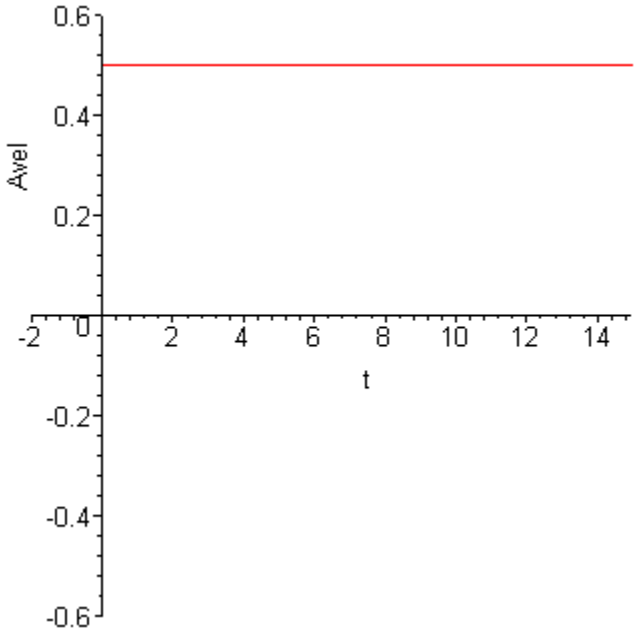
```
> timel:=16*Pi/3:
  theta:=t->-0.05*t^2:
  r:=t->5:
  x:=t->r(t)*cos(theta(t)):
  y:=t->r(t)*sin(theta(t)):
  rvectorCartesian:=[x(t),y(t)];
  plot([op(rvectorCartesian), t=0..timel], labels=[x,y]);
```

$$rvectorCartesian := [5\cos(0.05\,t^2), -5\sin(0.05\,t^2)]$$

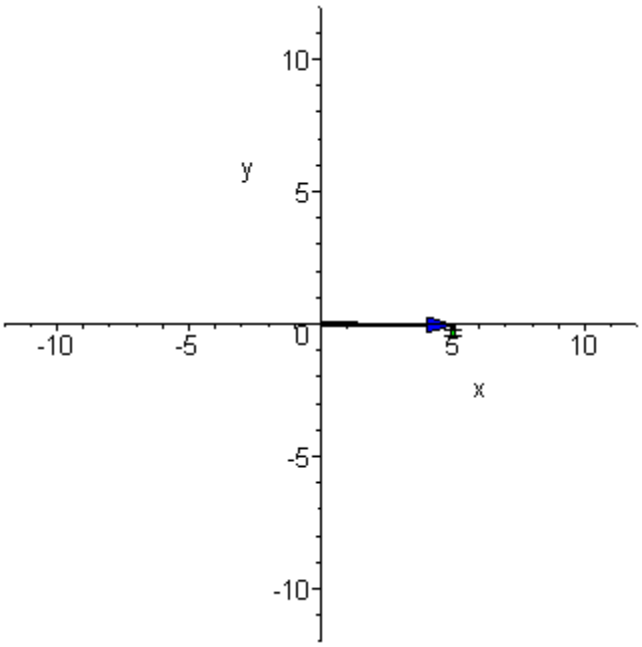



```
> window:=12:
  timel:=16*Pi/3:
```

```
vvectorCartesian:=diff(rvectorCartesian,t):
avectorCartesian:=diff(vvectorCartesian,t):
speed:=simplify(sqrt(vvectorCartesian[1]^2+vvectorCartesian[2]^2)):
vdirection:=[vvectorCartesian[1]/speed,vvectorCartesian[2]/speed]:
prod:=linalg[dotprod](avectorCartesian, vdirection):
plot(prod, t=0..15, labels=["t","Avel"],labeldirections=[HORIZONTAL,VERTICAL],view
2..15,-0.6..0.6]);
```




```
> for i from 1 to 40 do
  p[i]:=plot([op(rvectorCartesian), t=0..i*timel/40], labels=["x","y"]):
  a1:=x(i*timel/40):
  a2:=y(i*timel/40):
  a[i]:=arrow([0,0], [a1,a2], .1, .6, .2, color=blue):
  b1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, vvectorCartesian)):
  b[i]:=arrow([a1,a2], b1, .1,.6,.2, color=red):
  c1:=evalf(subs(t=i*timel/40,rvectorCartesian)+subs(t=i*timel/40, avectorCartesian)):
  c[i]:=arrow([a1,a2], c1, .1,.6,.2, color=green):
ani[i]:=display({p[i],a[i],b[i],c[i]}):
od:
display(seq(ani[i], i=1..40), insequence=true, view=[-window..window, -window..window]);
```


10
y
5
-10
-5
0
5
10
x
-5
-10

>