

Planetary Motion

Worksheet by Mike May, S.J. -©2006- maymk@slu.edu

```
[> restart;
```

One use of parameterized curves is to plot planetary motion. That is actually a good example to study since one could make a good case for the claim that Newton invented calculus to solve problems of planetary motion.

This is a demonstration worksheet without exercises.

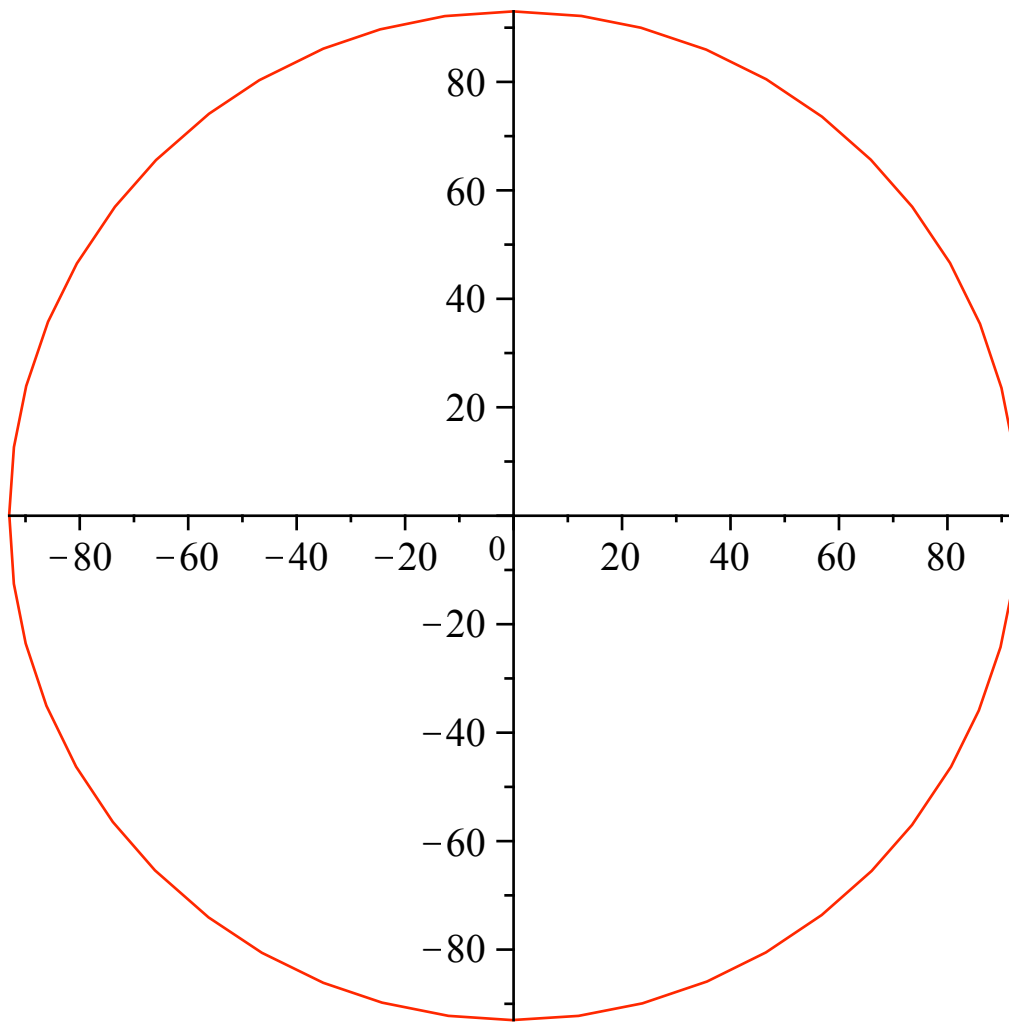
▼ Parameterizing elliptical motion

The first approach to planetary motion in the class is to look at trying to parameterize the elliptical paths of the planets. We want to start with parameterizing circular motion and modifying it.

If we want to move around a circle of radius r and period ω the easiest parameterization is $x(t) = a \cos(2\pi t/\omega)$, $y(t) = a \sin(2\pi t/\omega)$.

For the earth, $r=93$ in millions of miles, and $\omega = 1$ in years.

```
> a := 93: omeg := 1:  
x := t -> a*cos(2*Pi*t/omeg): y := t -> a*sin(2*Pi*t/omeg):  
plot([x(t), y(t), t=0..1] );
```



Next we want to take into account that the orbit of the earth is flattened into an ellipse. The eccentricity of the earth's orbit is .02.

That means that the sun is shifted $c = e \cdot a = .02 \cdot 93$ million miles off center.

It also means that the short axis is $b = \sqrt{a^2 - c^2} = a \cdot \sqrt{1 - e^2}$ million miles.

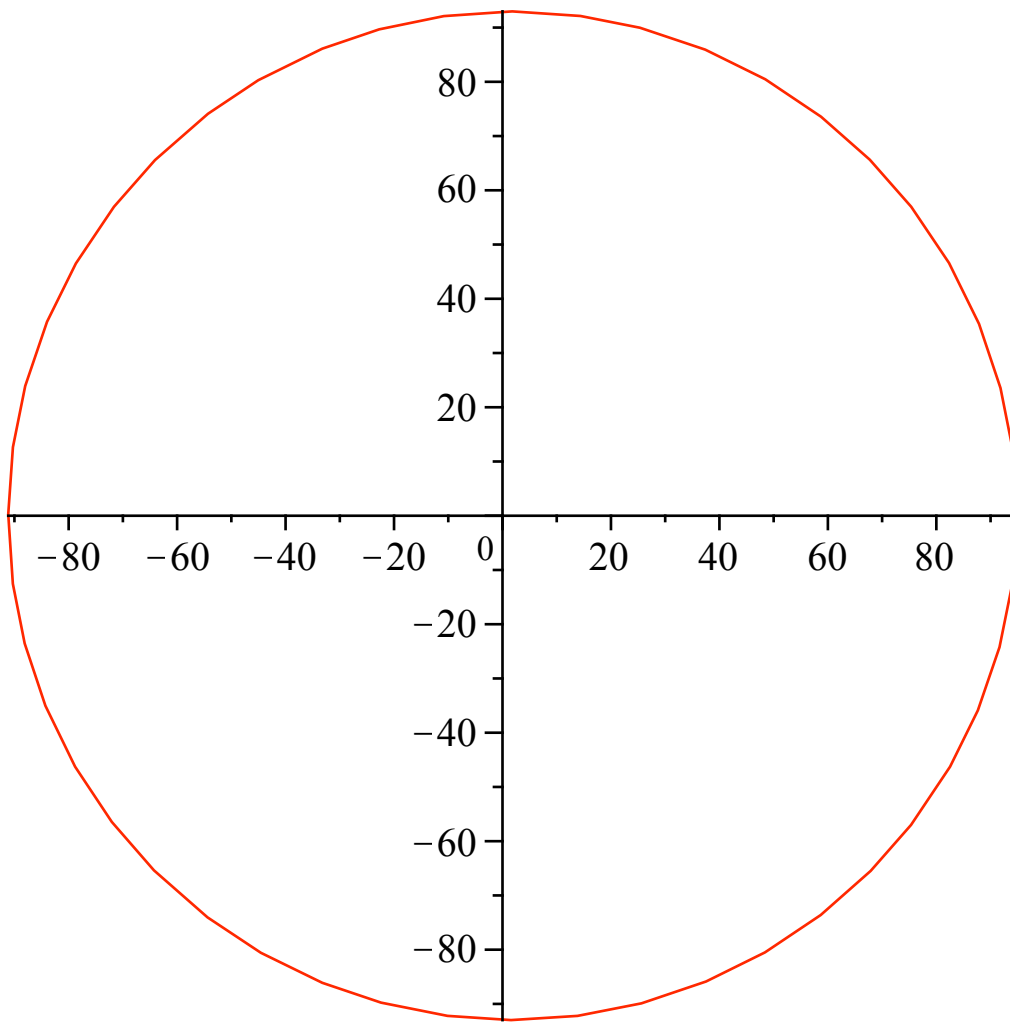
```
> e := .02:
   c := a*e;
   b := a*sqrt(1-e^2);
```

```
1.86
92.98139814
```

(1.1)

This leads to a new parameterization of the earth's motion.

```
> a := 93: omeg := 1:
   e := .02: c := a*e: b := a*sqrt(1-e^2):
   x := t -> a*cos(2*Pi*t/omeg) + c:
   y := t -> b*sin(2*Pi*t/omeg):
   plot([x(t), y(t), t=0..1] );
```

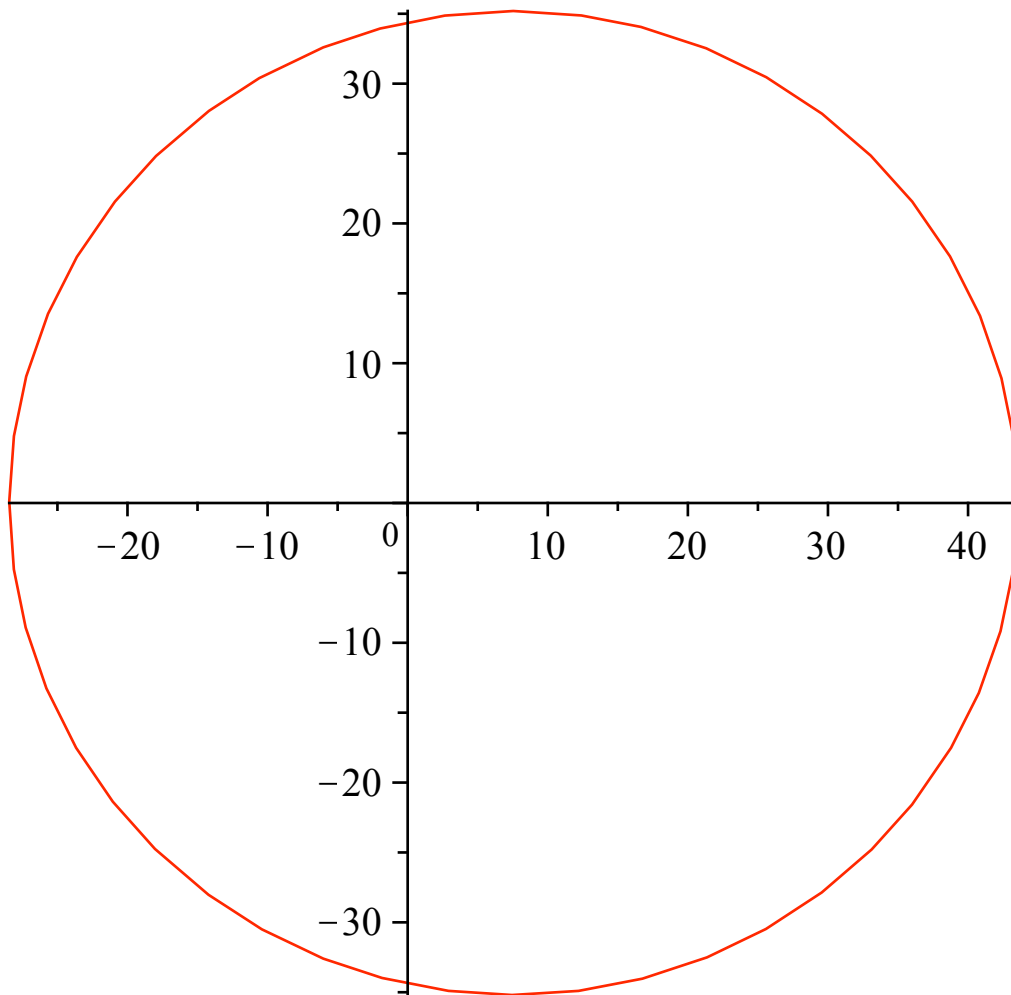


This still looks like a circle to the unaided eye.

If we look at the orbit of Mercury, a is 36 million miles and e is .21.

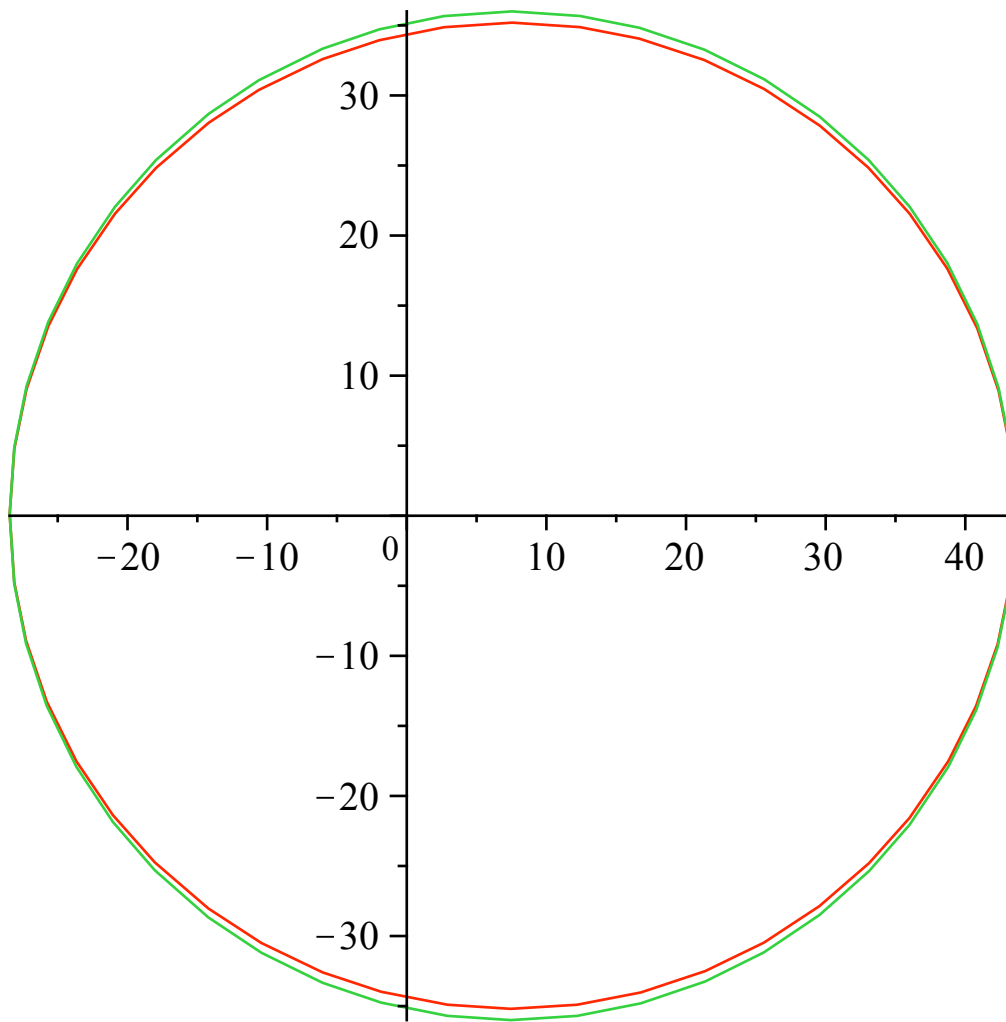
We also need to compute the period, which will be $(36/93)^{3/2}$. (The period is the size of the orbit, measured in AUs (The size of the earth's orbit) raised to the $3/2$ power.)

```
> a := 36:   omeg := evalf((a/93.0)^(3/2));
e := .21:   c := a*e:   b := a*sqrt(1-e^2):
x := t -> a*cos(2*Pi*t/omeg) + c:
y := t -> b*sin(2*Pi*t/omeg):
plot([x(t), y(t), t=0..omeg], scaling=constrained);
0.2408403938
```



For comparison we plot a circle at the same center.

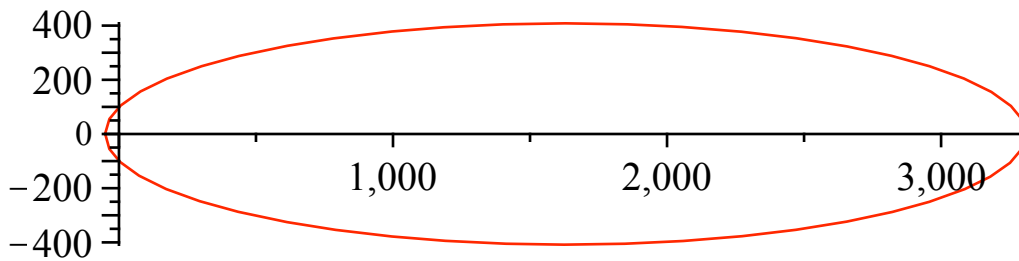
```
> xc := t -> a*cos(2*Pi*t/omeg) + c:  
yc := t -> a*sin(2*Pi*t/omeg):  
plot({[xc(t), yc(t), t=0..omeg], [x(t), y(t), t=0..omeg]},  
scaling=constrained);
```



It is still very close to circular in orbit, even if noticeably off center.

For Halley's comet the numbers are that $a=1680$ and $e=.97$.

```
> a := 1680:   omeg := evalf((a/93.0)^(3/2));
e := .97:     c := a*e:   b := a*sqrt(1-e^2):
x := t -> a*cos(2*Pi*t/omeg) + c:
y := t -> b*sin(2*Pi*t/omeg):
plot([x(t), y(t), t=0..omeg], scaling=constrained);
76.77847824
```



This orbit is clearly an ellipse. It is worth noting that our model would have the period of Mercury be 88 days and Haley's comet be 77 years.

[>

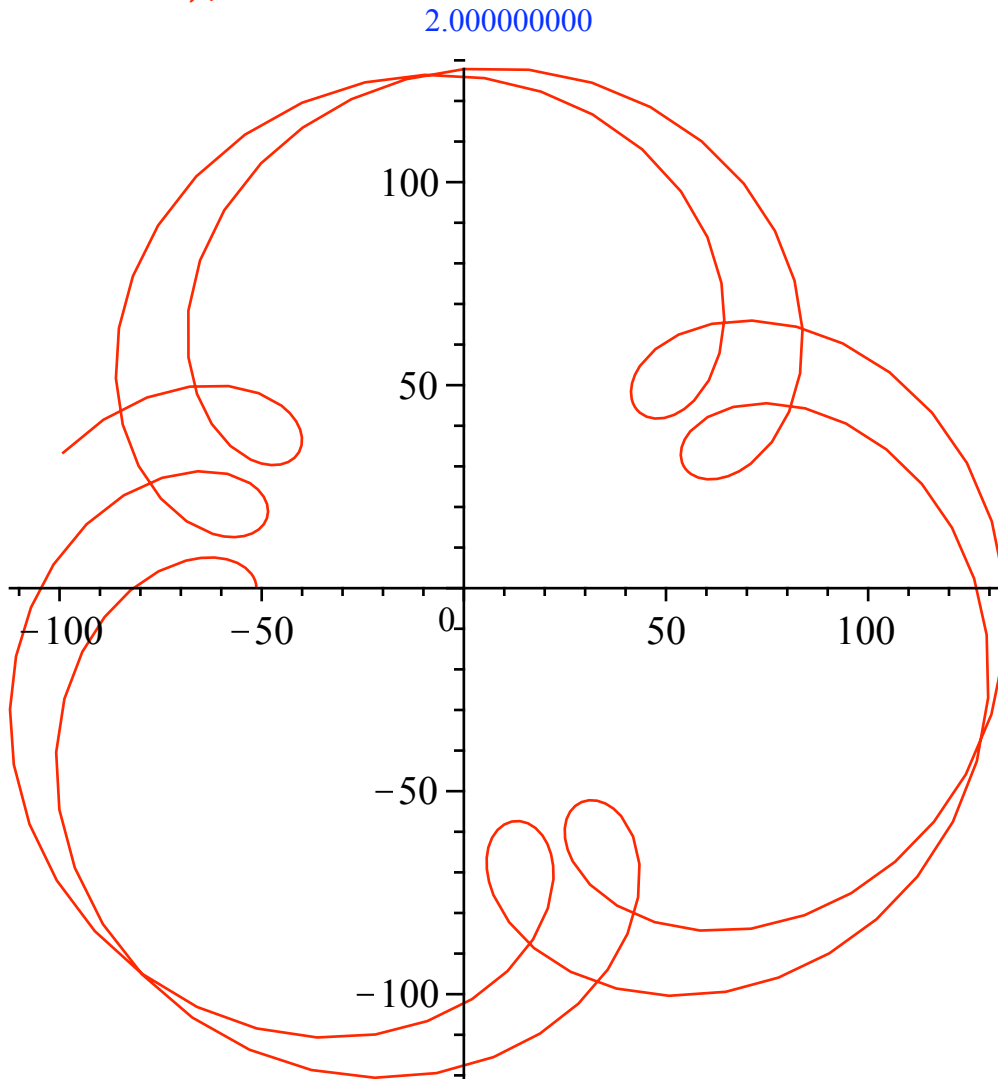
Relative motions

The next question of interest is to plot relative motion for two planets. In parametric form, we simply take the difference of the two parameterizations. Not that we want to take our time period to be at least the longer of the two periods.

Consider Mercury and the earth together.

```
> am := 36:   omegm := evalf((am/93.0)^(3/2)):
em := .21:   cm := am*em:   bm := am*sqrt(1-em^2):
xm := t -> am*cos(2*Pi*t/omegm) + cm:
ym := t -> bm*sin(2*Pi*t/omegm):
ae := 93:   omege := evalf((ae/93.0)^(3/2)):
ee := .02:   ce := ae*ee:   be := ae*sqrt(1-ee^2):
xe := t -> ae*cos(2*Pi*t/omege) + ce:
ye := t -> be*sin(2*Pi*t/omege):
tmax := max(2*omegm, 2*omege);
plot([xm(t) - xe(t), ym(t) - ye(t), t=0..tmax], scaling=
```

```
constrained);
```

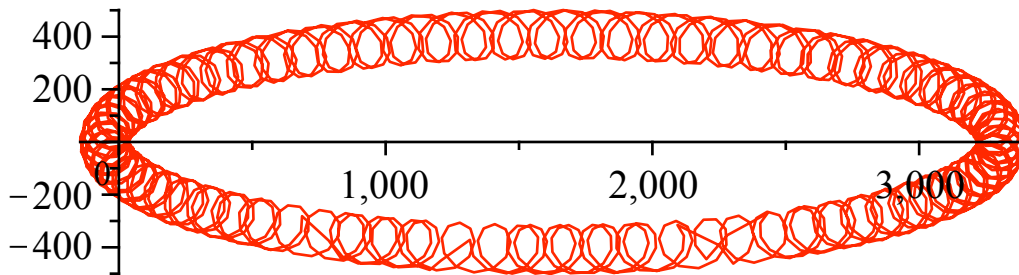


Looking from one planet to the other it appears that the planet is stopping and starting with some backing up as well.

Consider what happens if we try the earth and Halley's comet.

```
> am := 1680:   omegm := evalf((am/93.0)^(3/2)):
em := .97:    cm := am*em:    bm := am*sqrt(1-em^2):
xm := t -> am*cos(2*Pi*t/omegm) + cm:
ym := t -> bm*sin(2*Pi*t/omegm):
ae := 93:    omege := evalf((ae/93.0)^(3/2)):
ee := .02:   ce := ae*ee:    be := ae*sqrt(1-ee^2):
xe := t -> ae*cos(2*Pi*t/omege) + ce:
ye := t -> be*sin(2*Pi*t/omege):
tmax := max(2*omegm, 2*omege);
plot([xm(t) - xe(t), ym(t) - ye(t), t=0..tmax], scaling=
constrained);
```

153.5569565



The motion is obviously wrong. We have neglected that fact that planets move at different speeds on their orbits. They go faster when they are near the sun.

[>

▼ Finding a path determined by gravity

If we are to look at planetary motion from calculus, what we know is that the only force that acts on the planets is gravity and that it is in the direction of the sun with a force inversely proportional to the distance.

In terms of differential equations this becomes:

$$x'' = -x/r^3, \quad y'' = -y/r^3.$$

(Note that $x^2 + y^2 = r^2$ and that $\sqrt{(x'')^2 + (y'')^2} = 1/r^2$.)

We have only worked with vector fields that are first order differential equations and this system is second order. We handle that problem by defining $u=x'$ and $v=y'$ to get a system of 4 first order equations.

```
[ > x:='x': y:= 'y': u :='u': v:= 'v': t:= 't':
[ > ODEa := diff(x(t),t)=u(t):
  ODEb := diff(y(t),t)=v(t):
  ODEc := diff(u(t),t)=-x(t)/(x(t)^2+y(t)^2)^(3/2):
```

```

ODEd := diff(v(t),t)=-y(t)/(x(t)^2+y(t)^2)^(3/2):
eqns := [ODEa, ODEb, ODEc, ODEd];

```

$$\left[\begin{array}{l} \frac{d}{dt} x(t) = u(t), \frac{d}{dt} y(t) = v(t), \frac{d}{dt} u(t) = -\frac{x(t)}{(x(t)^2 + y(t)^2)^{3/2}}, \frac{d}{dt} v(t) = \\ -\frac{y(t)}{(x(t)^2 + y(t)^2)^{3/2}} \end{array} \right] \quad (3.1)$$

We then specify initial conditions and use the command dsolve to produce a solution function.

```

> ICx := x(0) = 1:
  ICy := y(0) = 0:
  ICu := u(0) = 0:
  ICv := v(0) = 1:
  solcurve := dsolve({ODEa, ODEb, ODEc, ODEd, ICx, ICy, ICu,
  ICv},
  [x(t), y(t), u(t), v(t)], numeric);
  proc(x_rkf45) ... end proc

```

(3.2)

Then we can look at where the planet is at times $\pi/2$, π , $3\pi/2$, and 2π .

```

> P1 := solcurve(Pi/2);
  P2 := solcurve(Pi);
  P3 := solcurve(3*Pi/2);
  P4 := solcurve(2*Pi);

```

$$\begin{aligned} [t=1.57079632679490, x(t) &= -4.87817260009203169 \cdot 10^{-7}, y(t) \\ &= 0.99999924677795660, u(t) = -1.00000028690824849, v(t) \\ &= -0.00000129483677848677044] \\ [t=3.14159265358979, x(t) &= -.999997747966016480, y(t) \\ &= -0.00000390191373064990160, u(t) = 0.00000479098866859164270, v(t) \\ &= -1.00000129387812442] \\ [t=4.71238898038468, x(t) &= 0.0000110542500780827328, y(t) \\ &= -.999996585358552736, u(t) = 1.00000198868719048, v(t) \\ &= 0.0000113844094413633260] \\ [t=6.28318530717958, x(t) &= 0.999996166659731456, y(t) \\ &= 0.0000198086280958540382, u(t) = -0.0000200848144339099265, v(t) \\ &= 1.00000191386769588] \end{aligned} \quad (3.3)$$

We are really interested in the values of x and y at those 4 times.

```

> P1a := [op(2, P1[2]), op(2, P1[3])];
  P2a := [op(2, P2[2]), op(2, P2[3])];
  P3a := [op(2, P3[2]), op(2, P3[3])];
  P4a := [op(2, P4[2]), op(2, P4[3])];

```

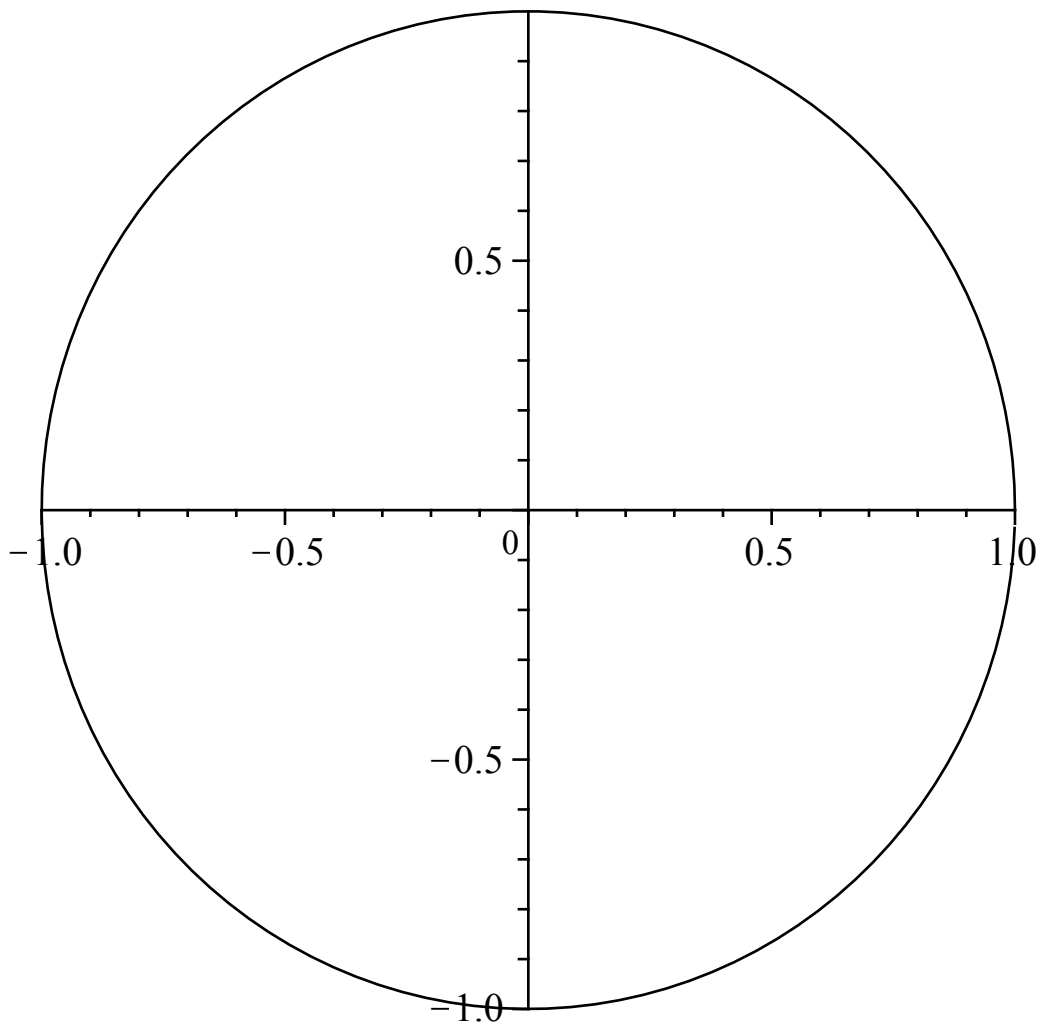
$$\begin{aligned} &[-4.87817260009203169 \cdot 10^{-7}, 0.99999924677795660] \\ &[-.999997747966016480, -0.00000390191373064990160] \\ &[0.0000110542500780827328, -.999996585358552736] \\ &[0.999996166659731456, 0.0000198086280958540382] \end{aligned} \quad (3.4)$$

This lets us see that with 6 decimal points of accuracy, the path is a circle with period 2π . We can graph the results to see this more clearly.

```

> with(plots):
  pointplot([seq([op(2, solcurve(t*.05)[2]), op(2, solcurve(t*.05)
  [3])],
  t=0..125)], connect=true);

```

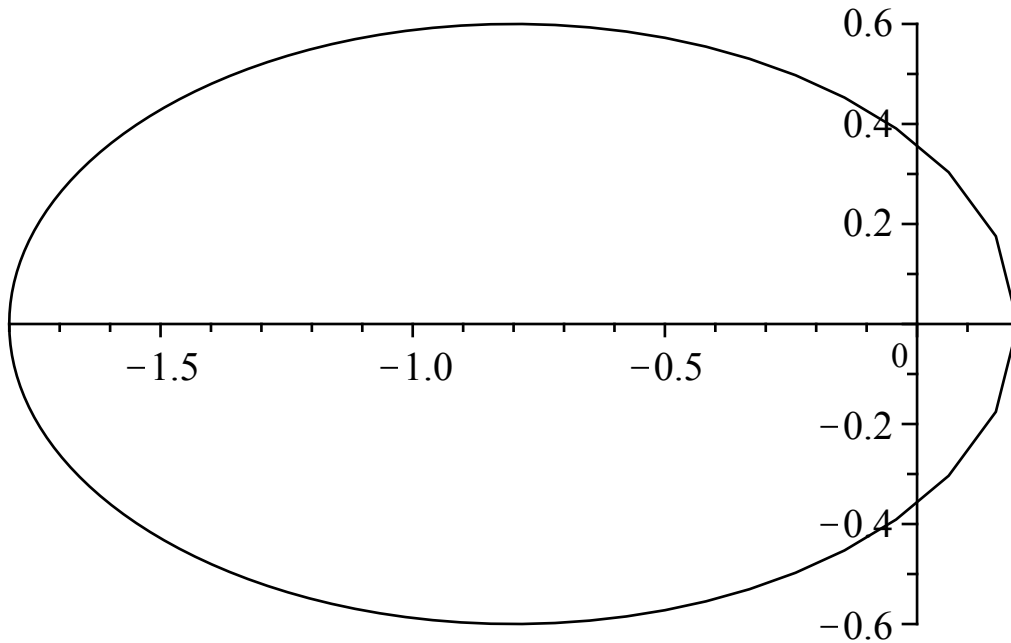


We can also see what happens if we change the eccentricity.

```

> e := 0.8; a := 1; tmax := 2*Pi*a*sqrt(a):
  ICx := x(0) = a*(1-e);
  ICy := y(0) = 0:
  ICu := u(0) = 0:
  ICv := v(0) = evalf(sqrt((1+e)/(a*(1-e))));
  solcurve := dsolve({ODEa, ODEb, ODEc, ODEd, ICx, ICy, ICu,
  ICv},
    [x(t), y(t), u(t), v(t)], numeric);
  pointplot([seq([op(2,solcurve(t*tmax/100)[2]),
    op(2,solcurve(t*tmax/100)[3])],
    t=0..100)], connect=true, scaling=constrained);
    0.8
    1
    x(0) = 0.2
    v(0) = 3.000000000
  proc(x_rkf45) ... end proc

```



We see that we get an ellipse for the orbit.

We are now ready to return to the question of relative orbits.

Consider first the problem of the earth and Mercury. We will measure in astronomical units.

```

> ee := 0.02; ae := 1; tmaxe := 2*Pi*ae*sqrt(ae);
  ICxe := x(0) = ae*(1-ee);
  ICye := y(0) = 0;
  ICue := u(0) = 0;
  ICve := v(0) = evalf(sqrt((1+ee)/(ae*(1-ee))));
  solcurvee := dsolve({ODEa, ODEb, ODEc, ODEd, ICxe, ICye,
    ICue, ICve},
    [x(t), y(t), u(t), v(t)], numeric);
  em := 0.21; am := 36/93; tmaxm := 2*Pi*am*sqrt(am);
  ICxm := x(0) = am*(1-em);
  ICym := y(0) = 0;
  ICum := u(0) = 0;
  ICvm := v(0) = evalf(sqrt((1+em)/(am*(1-em))));
  solcurvem := dsolve({ODEa, ODEb, ODEc, ODEd, ICxm, ICym,
    ICum, ICvm},
    [x(t), y(t), u(t), v(t)], numeric);

```

0.02

1

(3.5)

```

      2 π
      x(0) = 0.98
      v(0) = 1.020204061
      proc(x_rkf45) ... end proc
      0.21
      12
      31
      48
      961 π√93
      x(0) = 0.3058064516
      v(0) = 1.989158378
      proc(x_rkf45) ... end proc

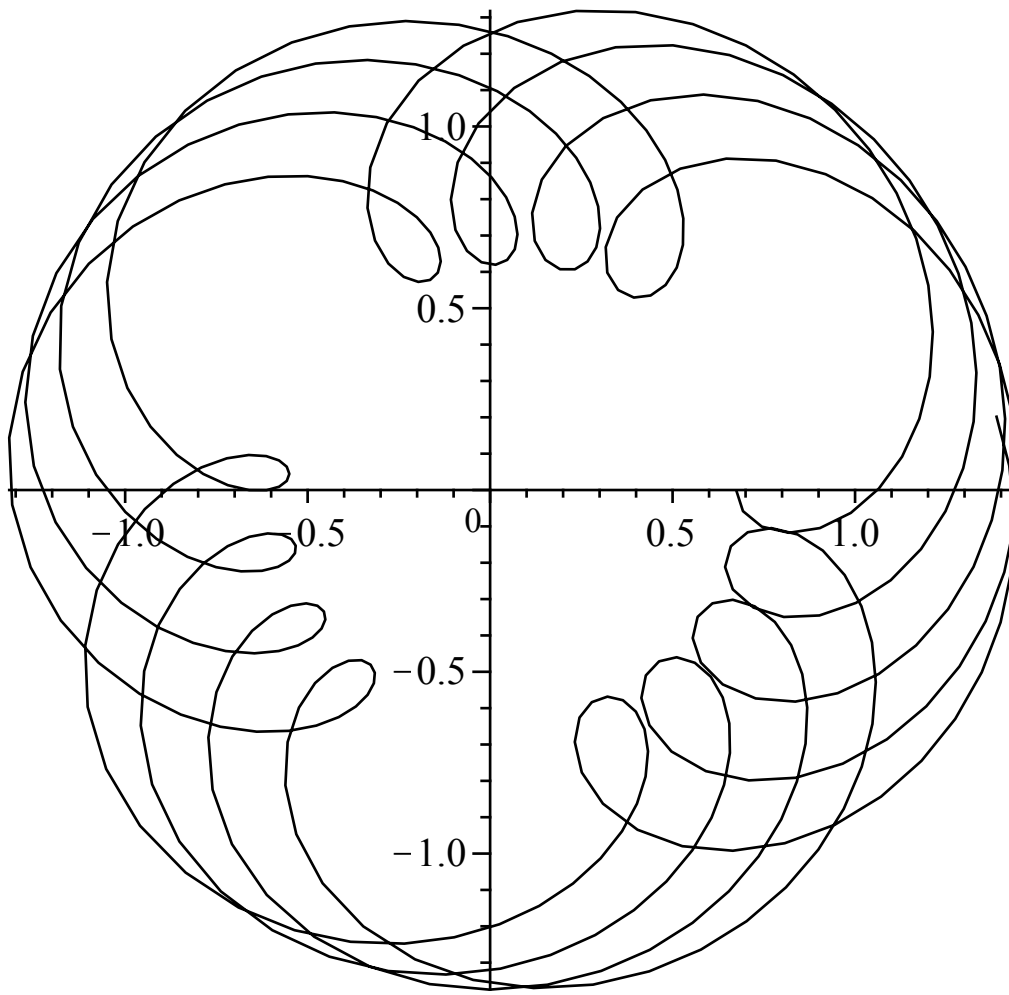
```

```

> tmax := 2*max(tmaxe, tmaxm);
pointplot([seq(
[op(2,solcurvee(t*tmax/200)[2])-op(2,solcurvem(t*tmax/200)[2]
),
op(2,solcurvee(t*tmax/200)[3])-op(2,solcurvem(t*tmax/200)[3]
],
t=0..400)], connect=true, scaling=constrained);

```

4 π

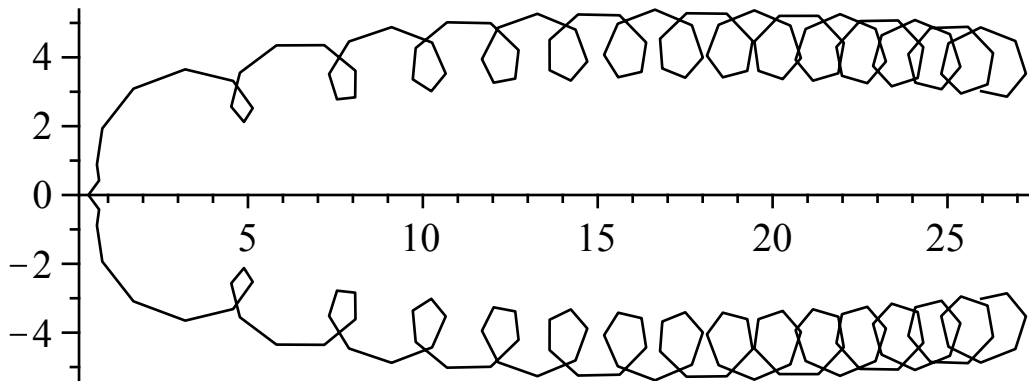


Notice the slight change in the relative path of the two planets. Some of the inner loops are clearly

smaller than others.

Now we try the same trick with Halley's comet.

```
> ee := 0.02; ae := 1; tmaxe := 2*Pi*ae*sqrt(ae);
  ICxe := x(0) = ae*(1-ee);
  ICye := y(0) = 0;
  ICue := u(0) = 0;
  ICve := v(0) = evalf(sqrt((1+ee)/(ae*(1-ee))));
  solcurvee := dsolve({ODEa, ODEb, ODEc, ODEd, ICxe, ICye,
  ICue, ICve},
    [x(t), y(t), u(t), v(t)], numeric);
  em := 0.97; am := 1680/93; tmaxm := evalf(2*Pi*am*sqrt(am));
  ICxm := x(0) = am*(1-em);
  ICym := y(0) = 0;
  ICum := u(0) = 0;
  ICvm := v(0) = evalf(sqrt((1+em)/(am*(1-em))));
  solcurvem := dsolve({ODEa, ODEb, ODEc, ODEd, ICxm, ICym,
  ICum, ICvm},
    [x(t), y(t), u(t), v(t)], numeric);
  tmax := 2*max(tmaxe, tmaxm);
  pointplot([seq(
  [op(2, solcurvee(t*tmax/1000)[2]) - op(2, solcurvem(t*tmax/1000)
  [2]),
  op(2, solcurvee(t*tmax/1000)[3]) - op(2, solcurvem(t*tmax/1000)
  [3])],
  t=-100..100)], connect=true, scaling=constrained);
    0.02
    1
    2 π
    x(0) = 0.98
    v(0) = 1.020204061
  proc(x_rkf45) ... end proc
    0.97
    560
    31
    482.4134062
    x(0) = 0.5419354839
    v(0) = 1.906598816
  proc(x_rkf45) ... end proc
    964.8268124
```

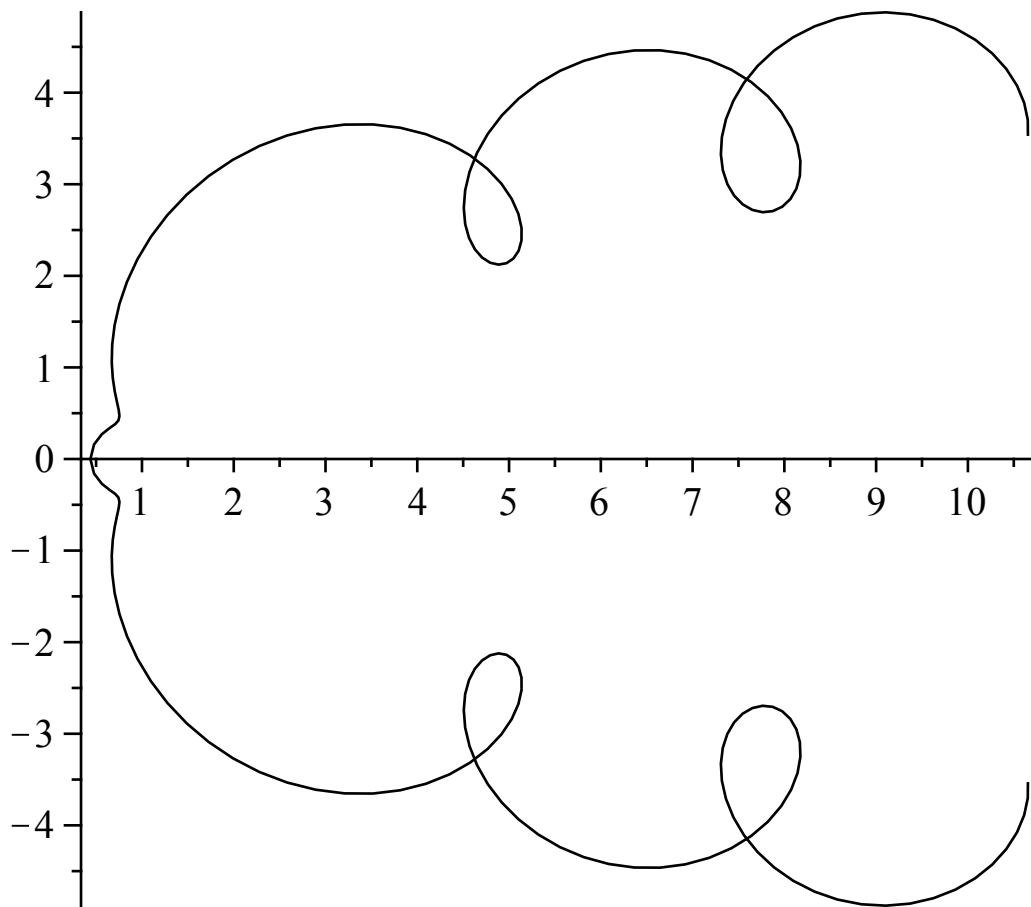


Some things to notice from the computations.

At its closest point, Halley's comet is inside the earth's orbit and moving almost twice as fast. If a year is 2π , then Halley's comet orbits once every $(482/(2\pi)) = 76.7$ years.

The part of the orbit close in is worth looking at in more detail.

```
> pointplot([seq(
  [op(2,solcurvee(t*tmax/5000)[2])-op(2,solcurvem(t*tmax/5000)
  [2]),
  op(2,solcurvee(t*tmax/5000)[3])-op(2,solcurvem(t*tmax/5000)
  [3])],
  t=-100..100)], connect=true, scaling=constrained);
```



The reason for the odd shape of the path is that we have the comet and the Earth closest to the sun at the same time and on the same side. This has the comet passing the earth on the inside.

If we have the earth and Haley's comet on different sides of the sun at closest approach, we get a smoother curve.

```
> pointplot([seq(
  [op(2,solcurvee(t*tmax/5000+Pi)[2])-op(2,solcurvem(t*
  tmax/5000)[2]),
  op(2,solcurvee(t*tmax/5000+Pi)[3])-op(2,solcurvem(t*
  tmax/5000)[3])],
  t=-100..100)], connect=true, scaling=constrained);
```

