16

Conic Sections

Why Study Conic Sections?

- **They are key geometrical entities**. Conics are the curves obtained by slicing through a double cone, but also appear as the loci derived from simple distance properties. These definitions make them central in analytic geometry.

- **Their reflective and focusing properties**. Parabolic reflectors focus incoming rays to a single point. Ellipses reflect between two foci. These properties are essential in telescope mirrors, satellite dishes, headlights, and microphones.

- **As simple models for real-world phenomena**. Projectile motion (with no air resistance) is parabolic. Certain wavefronts and stress curves have conic shapes.

- **To describe motion in classical mechanics**. Planetary motion, satellite trajectories, and comet paths all follow conic geometries.

16.1 Introduction

One of the fascinating aspects of conic sections – ellipses (including circles), parabolas, and hyperbolas – is their definability in so many different ways. There's

the original geometric interpretation as curves formed when a plane cuts a double cone at various angles, a locus-distance construction where each conic section is a set of points satisfying a slightly different distance property, the focus-directrix (or eccentricity) approach where the points are obtained using the ratio of the distances to a focus and directrix, as quadratic equations of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, versions in terms of trigonometric and hyperbolic parameters, and definitions using polar coordinates.

In this chapter, we'll also look at how Dandelin spheres (a 19th century geometric construction) can be used to prove the equivalence of the cone-based and focus-directrix approaches, how reflections are handled by ellipses and parabolas, and consider the beauty of confocal curves.

Conics are almost certainly covered by every textbook with a title containing a phrase similar to "analytic geometry". An excellent example is *Calculus and Analytical Geometry* by Thomas et. al [**TFW96**], in particular the 9th edition which includes an entire chapter on conic sections. We also recommend Yates' *A Handbook on Curves and their Properties* [**Yat47**] and the discussion of conic section reflection in chapter 5 of Higgins' *Mathematics for the Imagination* [**Hig02**].

16.2 A Geometric Definition

The original definition of conic sections by Menaechmus (c.375-325 BC) is based on planes intersecting a cone at varying angles, although Apollonius (c.200 BC) is credited with introducing the double cone construction. An ellipse cuts obliquely across one of the cone's nappes, a parabola is parallel to the cone's edge, and a hyperbola slices both nappes (see Fig. 16.1).

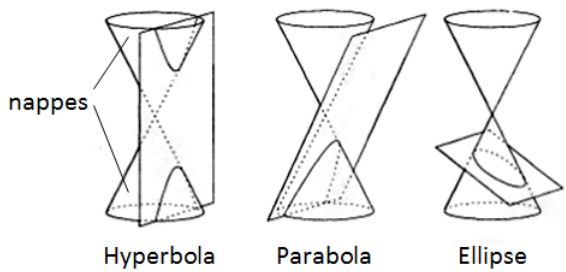


Figure 16.1. The Hyperbola, Parabola, and Ellipse

The circle is generally considered a variant of the ellipse, and there are three degenerate cases, shown in Fig. 16.2.

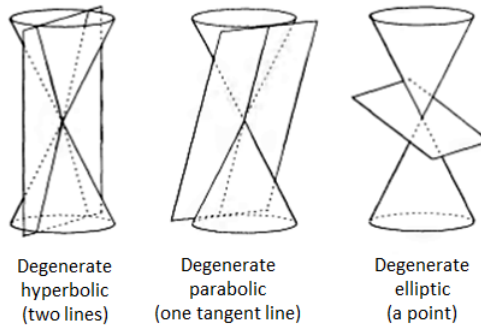


Figure 16.2. Degenerate Hyperbola, Parabola, and Ellipse

16.2.1 Classic Geometric Problems. Menaechmus probably first conceived of conic sections as a step to solving the three famous geometric problems of antiquity – trisecting an angle, doubling a cube, and squaring a circle (Fig. 16.3).

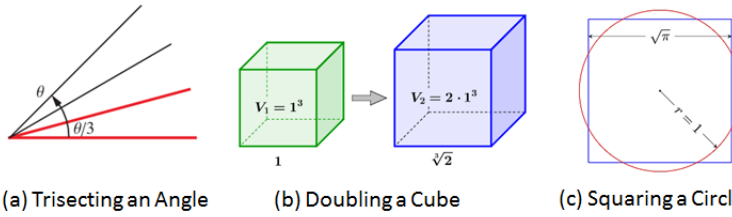


Figure 16.3. Three Classic Geometric Problems

For example, cube doubling requires the construction of an edge of a second cube whose volume is double that of the first. This can be addressed using conic sections in a variety of ways, as suggested by Fig. 16.4. Using a parabola and a hyperbola, as in Fig. 16.4(a), was Menaechmus's solution.

Famous Problems of Geometry and How to Solve Them by Benjamin Bold is an excellent brief introduction to these problems through the use of exercises [**Bol12**], while *A History of Mathematical Impossibility* by Jesper Lützen gives a historical presentation [**Lüt22**].

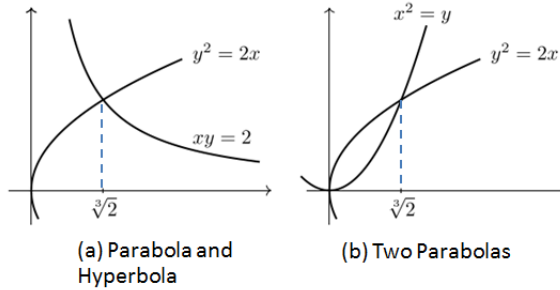


Figure 16.4. Two Cube Doublings Using Conics Sections

16.3 A Locus-Distance Definition

The locus-distance definition of conic sections is usually grouped with the focus-directrix approach described in the next section, but is a little simpler (not requiring directrixes for ellipses and hyperbolas), and suggests ways that these curves can be drawn. The equations are summarized in Fig. 16.5.

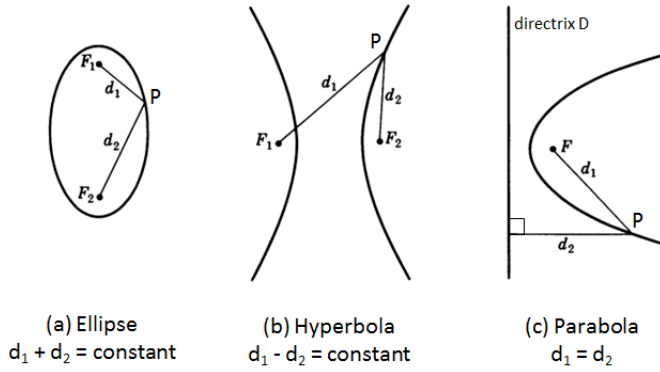


Figure 16.5. Locus/Distances for the Ellipse, Hyperbola, and Parabola

16.3.1 Parabolas. A parabola consists of all the points equidistant from a fixed point (the focus F) and a fixed line (the directrix D). In Fig. 16.5(c), the parabola is formed from the points P where $d_1 = d_2$.

More algebraically, suppose that the focus is the point $F(0, p)$ on the positive y-axis and that the directrix is the line $y = -p$ (Fig. 16.6).

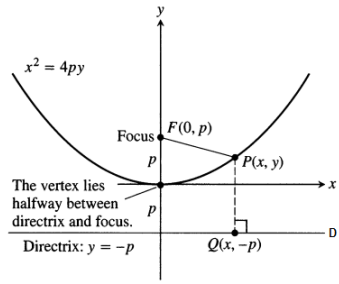


Figure 16.6. Locus/Distances for the Parabola

$P(x, y)$ lies on the parabola if and only if $PF = PQ$, which permits the distance formula to be written as

$$\begin{aligned} PF &= \sqrt{(x-0)^2 + (y-p)^2} = \sqrt{x^2 + (y-p)^2} \\ PQ &= \sqrt{(x-x)^2 + (y-(-p))^2} = \sqrt{(y+p)^2} \end{aligned}$$

We equate these expressions, square, and simplify to get

$$y = \frac{x^2}{4p} \quad \text{or} \quad x^2 = 4py$$

We can obtain similar equations for parabolas opening downwards, to the right, and to the left.

16.3.2 Ellipses. An ellipse is the set of points whose distances from two fixed points (its foci, F_1 and F_2) have a constant sum. In Fig. 16.5(a), the ellipse is formed from the points P where $d_1 + d_2 =$ a constant. Incidentally, the figure suggests a simple way to draw an ellipse – by tying a string to F_1 and F_2, and moving a pencil held tight against the string around the foci. We'll return to the issue of drawing conic sections using the locus-distance relations at the end of this section.

Suppose that the the foci are $F_1(-c, 0)$ and $F_2(c, 0)$ (Fig. 16.7), and $PF_1 + PF_2 = 2a$, then the coordinates of P on the ellipse satisfy the equation

$$\sqrt{(x+c)^2 + y^2} + \sqrt{(x-c)^2 + y^2} = 2a$$

Move the second square root to the right-hand side, square, isolate the remaining root, and square again, to obtain:

$$\frac{x^2}{a^2} + \frac{y^2}{a^2 - c^2} = 1.$$

Since $PF_1 + PF_2$ is greater than the length F_1F_2 (due to the triangle inequality), then $2a$ is greater than $2c$. Accordingly, $a > c$ and $a^2 - c^2$ in the above equation is positive.

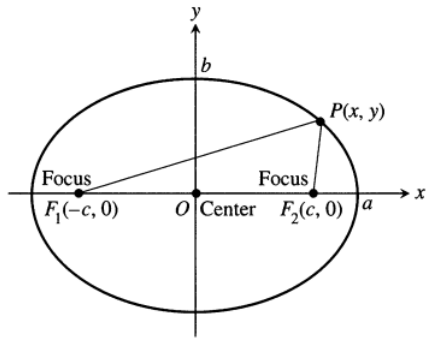


Figure 16.7. Locus/Distances for the Ellipse (algebraically)

Often we define $b = \sqrt{a^2 - c^2}$, where $(0, \pm b)$ are the points where the ellipse cross the y-axis; the equation becomes

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

The ellipse's *major axis* is the line segment joining the points $(\pm a, 0)$. The *minor axis* is the line segment joining the points $(0, \pm b)$. a is called the semi-major axis, b the semi-minor axis, and c the center-to-focus distance.

16.3.3 Hyperbola. A hyperbola (see Fig. 16.5(b)) is the set of points whose distances from two focus points have a constant *difference*. If the foci are $F_1(-c, 0)$ and $F_2(c, 0)$ (Fig. 16.8) and the constant difference is $2a$, then a point $P(x, y)$ lies on the hyperbola if and only if

$$\sqrt{(x + c)^2 + y^2} - \sqrt{(x - c)^2 + y^2} = \pm 2a.$$

Move the second square root to the right-hand side, square, isolate the remaining root, and square again, to produce:

$$\frac{x^2}{a^2} + \frac{y^2}{a^2 - c^2} = 1.$$

This looks uncannily like the equation for an ellipse, but $a^2 - c^2$ is negative because $2a$, being the difference of two sides of $\triangle PF_1F_2$, is less than $2c$, the third side.

If we let b be the positive square root of $c^2 - a^2$, then $a^2 - c^2 = -b^2$, and the equation becomes

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1.$$

Note that the hyperbola crosses the x-axis at the points $(\pm a, 0)$ (labeled as A_1 and A_2), and the hyperbola's constant difference $(d_1 - d_2)$ is $2a$. In other words,

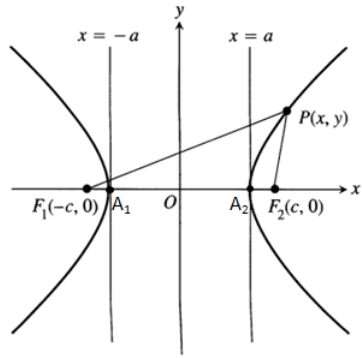


Figure 16.8. Locus/Distances for the Hyperbola (algebraically)

$2a$ is also the shortest perpendicular distance between the two branches of the hyperbola. Also, the distances F_1A_1 and F_2A_2 have the same $|c - a|$ value.

The hyperbola has two asymptotes, the lines

$$y = \pm\frac{b}{a}x.$$

One practical use for these lines is to help us graph the curve, as suggested by Fig. 16.9.

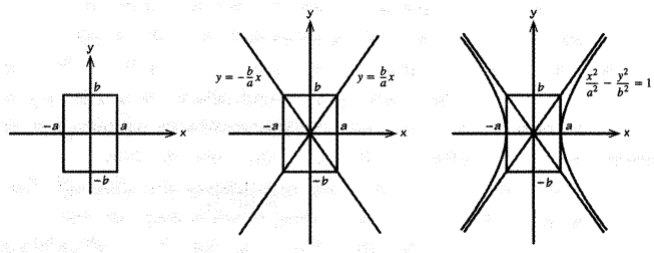


Figure 16.9. Sketching $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$ using Asymptotes

A quick way to find the asymptotes is to replace the 1 in the hyperbolic equation by 0 and solve for y:

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \;\Rightarrow\; \frac{x^2}{a^2} - \frac{y^2}{b^2} = 0 \;\Rightarrow\; y = \pm\frac{b}{a}x$$

|ł{z|}
0 for 1

16.3.4 Drawing Conics Sections. The locus-distance definitions for conic sections lend themselves to drawings using string, pins, and other simple tools. Fig. 16.10 illustrates the techniques.

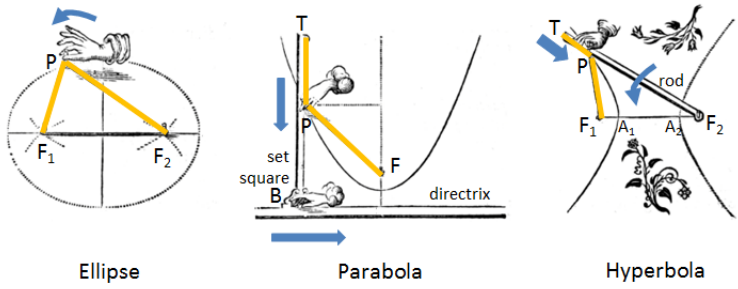


Figure 16.10. Locus/Distances Approaches for Drawing Conic Sections

The embellishments in Fig. 16.10 are due to our use of images from Frans van Schooten's *De Organica Conicarum Sectionum in Plano Descriptione Tractatus* textbook (1646). An animated version of these mechanisms can be found in "MathLapse: Constructions by pin-and-string: conics" at `https://www.youtube.com/watch?v=mldZ_7QwLvs`.

The drawing technique for the ellipse is fairly obvious, although one subtlety is that the string must be kept taut. There's a similar, but more complex requirement for the parabola: the string must be held tight against the line TP perpendicular to the directrix, and PF must also stay taut. The simplest way to see that this device actually works is to assume that the string is the same length as the set squares' height, which means that PB will always equal PF.

The sketching of a hyperbola uses a string whose length L_s equals that of the rod L_r minus the distance A_1A_2 (the shortest distance between the arms of the hyperbola:

$$L_s = L_r - A_1A_2 \quad \text{or} \quad L_r - L_s = A_1A_2$$

Recall that A_1A_2 is also the difference between the two foci distances: $d_2 - d_1 = A_1A_2 = 2a$.

Attach the rod to one of the foci (e.g. F_2 in Fig. 16.10(c)) so it can turn freely around the focus. Fasten the string to the other end of the rod (at T) and to the other focus (F_1). The user turns the rod around F_2 keeping the string tight by holding the pencil against the bar. We see that:

$$TP + PF_1 = L_s \quad \text{and} \quad TP + PF_2 = L_r$$

and so:

$$PF_2 - PF_1 = L_r - L_s = A_1A_2$$

As the rod turns, PF_1 and PF_2 change by the same amount, which means that $PF_2 - PF_1$ remains constant, or equivalently that $d_2 - d_1$ stays constant.

Although we've used van Schooten's drawings from 1620, it's worth mentioning the first printed popular textbook on geometric drawings was Albrecht Dürer's "Course in the Art of Measurement with Compass and Ruler" (*Underweysung der Messung mit dem Zirckel und Richtscheyt*), published in 1525. Aside from locus-direction drawings of conic sections, he also discussed point-by-point constructions and mechanical devices such as the Trammel of Archimedes. In other parts of the book, Dürer wrote about curves such as helices, conchoids, and epicycloids, how to draw polygons, and introduced the theory of perspective. A very nice overview of Dürer's contributions can be found in chapter 2 of Daniel Pedoe's *Geometry and the Liberal Arts* (later editions replace the word 'Liberal' with 'Visual') [**Ped83**].

16.3.5 Locus-Direction in Code. We've written three programs that reproduce the effect of drawing a parabola, ellipse, and hyperbola using the locus-direction approach: `paraString.py` (Fig. 16.11), `ellipseString.py`, and `hyperbolaString.py` (Fig. 16.12).

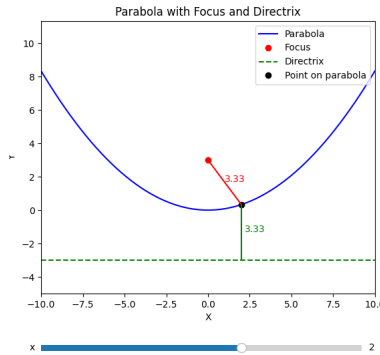
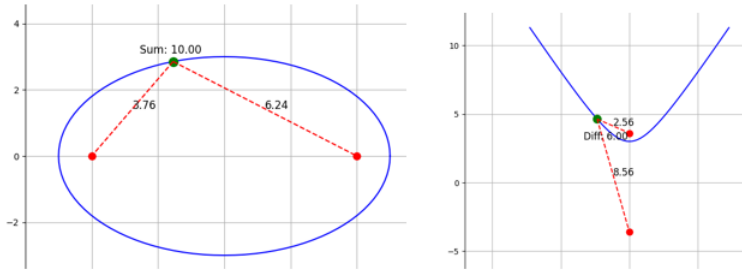


Figure 16.11. Drawing Locus/Distances for a Parabola

`paraString.py` lets the user slide a point along a parabola, and the distances between that point and the parabola's focus and directrix are reported. Relevant code snippets:

```
def parabola(x, d):
  return (x * x) / (4 * d)

d = 3.0  # directrix is y = -d
focus = (0, d)
# many lines not shown
```



(a) ellipseString.py (b) hyperbolaString.py

Figure 16.12. Drawing Locus-Distances for an Ellipse and a Hyperbola

```
    :
# draw curve, focus and directrix
ax.plot(xs, ys, color='blue', label='Parabola')
ax.plot([focus[0]], [focus[1]], 'ro', label='Focus')
ax.axhline(-d, color='green', ls='--', label='Directrix')
```

The slider returns an x-coordinate (`px`) which is used to calculate a parabola position, and its distances from the focus and directrix:

```
py = parabola(px, d)
distFocus = math.hypot(px - focus[0], py - focus[1])
distDirectrix = abs(py - (-d))
```

Reassuringly, the two distances, (`distFocus` and `distDirectrix`, remain equal throughout.

ellipseString.py and hyperbolaString.py use similar animation code to move a point repeatedly along an ellipse and (the upper half of) a hyperbola.

Relevant code fragments for the ellipse:

```
def ellipse(theta):
  x = semiMajor * math.cos(theta)
  y = semiMinor * math.sin(theta)
  return (x,y)

# ellipse parameters
semiMajor = 5  # axes lengths (a and b)
semiMinor = 3
c = math.sqrt(semiMajor**2 - semiMinor**2)
foci1 = (-c, 0); foci2 = (c, 0)
```

`ellipse()` is implemented using a parametric equation which we'll explain in section 16.10.3.

Matplotlib calls `animate()` to draw each frame of the animation. It converts the frame number into an angle which is passed to `ellipse()` to get the current

position of the moving point. The distances of that point from the two foci are reported (`len1` and `len2`), along with the sum (`sumLen`):

```
# animation for frame i
theta = 2*math.pi * i/100
xDot, yDot = ellipse(theta)
# many lines not shown
    :
len1 = math.sqrt((xDot - foci1[0])**2 + (yDot - foci1[1])**2)
len2 = math.sqrt((xDot - foci2[0])**2 + (yDot - foci2[1])**2)
sumLen = len1 + len2
```

Pleasingly, `sumLen` always retains the same value, 10, which is 2 × `semiMajor` ($2a$).

In hyperbolaString.py, the important parts of the code are:

```
def hyperbola(theta):
  x = semiMinor * math.sinh(theta)
  y = semiMajor * math.cosh(theta)
  return (x,y)

# vertical hyperbola parameters
semiMajor = 3  # axes lengths (a and b)
semiMinor = 2
c = math.sqrt(semiMajor**2 + semiMinor**2)
foci1 = (0, -c); foci2 = (0, c)
```

`hyperbola()` is implemented using a parametric equation that we'll explain in section 16.10.4.

Matplotlib calls `animate()` to draw each frame. The frame number is converted into an angle, and passed to `hyperbola()` to get the point's current position. The point's distances from the two foci are reported, (`len1` and `len2`), along with their absolute difference (`diffLen`):

```
# animation for frame i
theta = (i - 20)/10
xDot, yDot = hyperbola(theta)
# many lines not shown
    :
len1 = math.sqrt((xDot - foci1[0])**2 + (yDot - foci1[1])**2)
len2 = math.sqrt((xDot - foci2[0])**2 + (yDot - foci2[1])**2)
diffLen = abs(len1 - len2)
```

`diffLen` always reports the same value, 6, which is 2 × `semiMajor` ($2a$).

16.4 Classifying Conic Sections by Eccentricity

We have Pappus (c.320 AD) to thank for the focus–directrix definition of conics, which appeared in his multi-volume *Collection*, specifically Book VII, which describes the works of Euclid, Apollonius, Aristaeus, Eratosthenes, and others.

The *Collection* as a whole is a compendium of topics that were part of the classical mathematics curriculum, including geometry, astronomy, and mechanics. It had a significant impact on 17th century mathematics after being translated into Latin in 1588. For instance, it inspired Descartes' work on analytic geometry, specifically the quadratic equation treatment of conic sections that we'll meet later.

The focus–directrix definition states that $\frac{PF}{PD} = e$, where P is any point on the conic section, F a focus, D a directrix, and e (the eccentricity) is constant. The type of a conic section can be determined solely by considering the value of e:

- a parabola if $e = 1$,
- an ellipse if $e < 1$, and
- a hyperbola if $e > 1$.

Note that the parabola's definition is essentially unchanged from its locus-distance expression, but by utilizing eccentricity all of the conic sections can be formulated in a uniform way.

For the ellipse and hyperbola:

$$\texttt{Eccentricity} = \frac{\texttt{distance between foci}}{\texttt{distance between vertices}}$$

A more modern expression is

$$e = \frac{c}{a}.$$

This is equivalent to the more wordy definition since the distance between the foci is $2c$ and the distance between the vertices is $2a$ (see Fig. 16.13and Fig. 16.15).

Another version is:

$$PF_1 = e \cdot PD_1, \quad PF_2 = e \cdot PD_2$$

where F_1 and F_2 are the foci and D_1 and D_2 are the directrixes.

16.4.1 Ellipse. There are several ways to visualize e for an ellipse, as shown in Fig. 16.13.

The equivalence of

$$PF_1 = e \cdot PD_1, \quad PF_2 = e \cdot PD_2$$

and $e = \frac{c}{a}$ is easiest to see by considering when P is at a vertex, between a focus and its directrix (see Fig. 16.14).

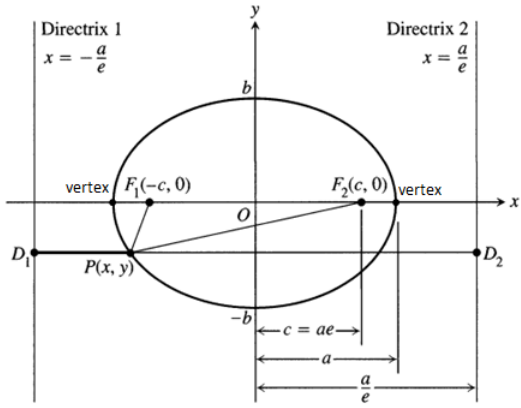


Figure 16.13. An Ellipse's Eccentricity

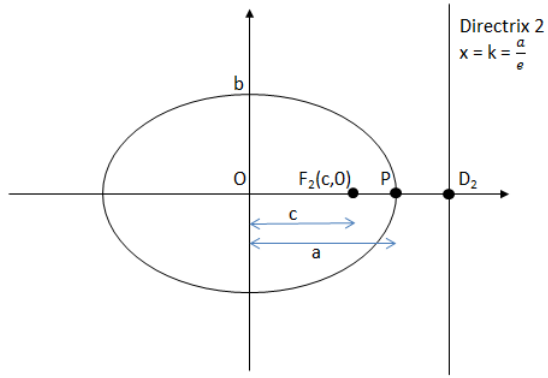


Figure 16.14. An Ellipse with P at a Vertex

In that case

$$\begin{aligned} e &= \frac{PF_2}{PD_2} = \frac{a-c}{(a/e)-a} \\ e((a/e)-a) &= a-c \\ a-ae &= a-c \\ ae &= c \\ e &= \frac{c}{a} \end{aligned}$$

If we fix a and vary c over the interval $0 \leq c \leq a$, then the ellipse will be circular when $c = 0$ (which means that $a = b$) and flattens as c increases. If $c = a$, the ellipse degenerates into a line segment.

16.4.2 Hyperbola. The eccentricity of the hyperbola $(x^2/a^2) - (y^2/b^2) = 1$ is:

$$e = \frac{c}{a} = \frac{\sqrt{a^2 + b^2}}{a}$$

As with the ellipse, there are several ways to visualize e, as shown in Fig. 16.15.

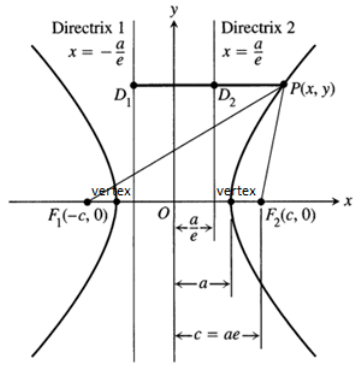


Figure 16.15. A Hyperbola's Eccentricity

16.5 Polar Coordinates

Every Cartesian point P can be assigned a polar coordinate (r, θ), where r is the distance from the origin O to P, and θ is the angle measured counterclockwise from the initial direction (typically the positive x-axis) to OP. This angle isn't unique, so a Cartesian point can have infinitely many polar coordinates. For instance, a point 2 units from the origin at $\theta = \pi/6$ has the polar coordinate $P(2, \pi/6)$, and also $P(2, -11\pi/6)$, $P(2, 13\pi/6)$, etc, in multiples of $\pm 2\pi$.

As a further source of ambiguity, r may sometimes be negative. For example, $P(2, 7\pi/6)$ can be reached by turning $7\pi/6$ radians counterclockwise and going forward 2 units, but it can also be located by turning only $\pi/6$ radians counterclockwise and going backward 2 units, resulting in $P(-2, \pi/6)$.

16.6 Ellipses, Parabolas, and Hyperbolas Unified

To formulate polar equations for the ellipse, parabola, and hyperbola, we place one focus at the origin and its directrix to the right of the origin at $x = k$ (Fig. 16.16).

$PF = r$ and $PD = k - FB = k - r\cos\theta$, which means that the conic section's focus-directrix equation $PF = e \cdot PD$ becomes

$$r = e(k - r\cos\theta),$$

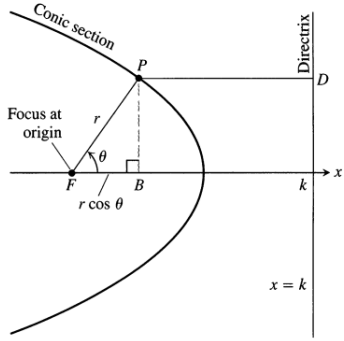


Figure 16.16. A Polar Equation for a Conic Section

which can be rearranged as:

$$r = \frac{ke}{1 + e\cos\theta}.$$

This represents an ellipse if $0 < e < 1$, a parabola if $e = 1$, and a hyperbola if $e > 1$. In other words, polar coordinates and eccentricities let us define all of the conic sections with a single equation.

You may see variations of this equation depending on the location of the directrix. For instance, if the directrix is $x = -k$ to the left of the origin, we get

$$r = \frac{ke}{1 - e\cos\theta}.$$

If the directrix is relative to the y-axis (as $y = k$ or $y = -k$ in Fig. 16.17), then the equation will use sines instead of cosines.

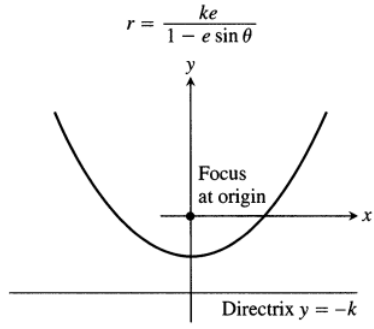


Figure 16.17. A Polar Sine Equation for a Conic Section

One variation is to replace k by the semi-major axis a. This is easy but we must remember that the polar equations so far have assumed the focus is at the origin, which is not the case in Fig. 16.13 and Fig. 16.15. In Fig. 16.13,

$$k = \frac{a}{e} - ea.$$

which can be rearranged as $ke = a(1 - e^2)$. Then the polar equation becomes:

$$r = \frac{a(1 - e^2)}{1 + e\cos\theta}$$

Notice that when $\theta = 0$, this equation becomes $r = a$, which represents a circle.

Yet another alternative is to utilize the conic section's *latus rectum*. This is the line segment that passes through a focus, perpendicular to the major axis, with its endpoints on the curve (Fig. 16.18). The *semi*-latus rectum, l, is half of the latus rectum, representing the distance from the focus to one of the line's endpoints.

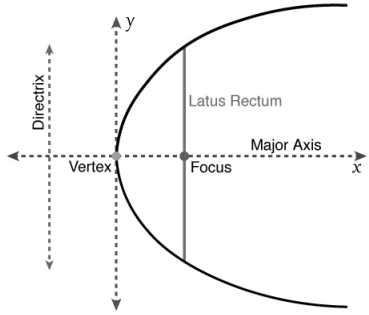


Figure 16.18. The Latus Rectum of a Conic Section

For the parabola $y^2 = 4ax$, $l = 2a$, while for the ellipse and hyperbola, $l = b^2/a$. However, since we're dealing with eccentricity, e, and a $x = \pm k$ directrix, it's more useful to express it as $l = ke$. This gives us the polar equation:

$$r = \frac{l}{1 + e\cos\theta}.$$

16.7 Drawing Conic Sections using their Eccentricity

We've implemented two programs, conicsEccen.py and conicsPolar.py, that draw conic sections with polar coordinates.

`conicsEccen.py` employs $r = l/(1 + e\cos(\theta))$ but keeps l constant ($l = 1$), and fixes one focus at the origin. However, e is adjustable via a slider, which

permits the curve to change between a circle, ellipse, parabola, and hyperbola. As it transforms, the directrix associated with the fixed focus moves, and a second focus and directrix will appear and disappear.

The second focus and directrix move to the left as the eccentricity grows from 0 towards 1, eventually reaching $-\infty$ as the ellipse becomes a parabola (see Fig. 16.19). As the eccentricity increases above 1, the eccentricity returns from $+\infty$ as a hyperbola. The focus and directrix reappear on the right of the window as the right hand branch of the hyperbola moves towards its left branch.

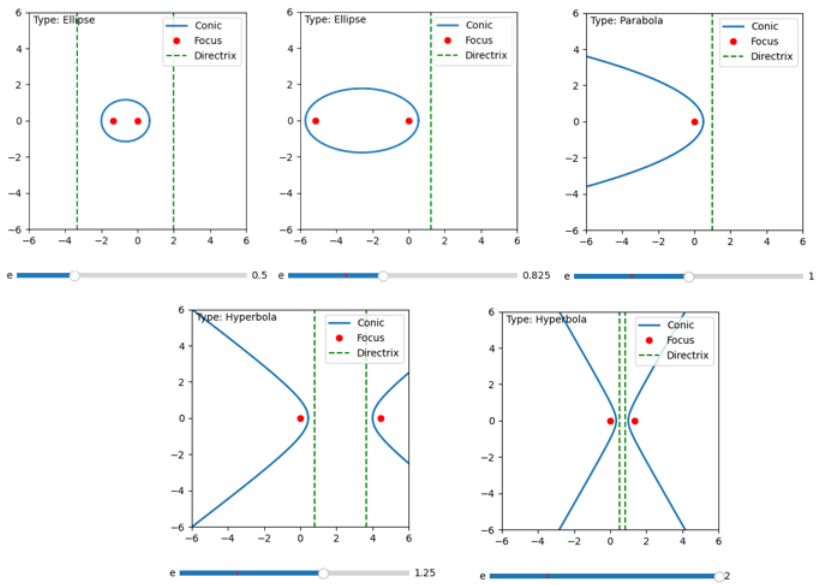


Figure 16.19. `conicsEccen.py`: increasing e

By referring to Fig. 16.13 and Fig. 16.15, it's clear that the second focus is at the same distance from the center at the fixed focus at the origin, and so is positioned at $(\pm 2c, 0)$ depending on whether the curve is an ellipse or hyperbola. The distance between the fixed focus and its directrix is k, a distance also used by the second focus and directrix. So the second directrix will be at $x = \pm 2c - k$. Since $l = ke$, this can be rewritten as $x = \pm 2c - l/e$, which is preferable since l is constant.

`conicType()` in conicsEccen.py converts e into a string:

```
def conicType(e):
  if abs(e) < 1e-6:  # v.close to 0
    return "Circle"
```

```
  elif e < 1:
    return "Ellipse"
  elif abs(e - 1) < 1e-6:  # v.close to 1
    return "Parabola"
  else:
    return "Hyperbola"
```

This string is used in `update()` which redraws the plot whenever the slider is moved. The following code fragment shows how the second focus (`focus2Dot`) and directrix (`drx2`) are calculated:

```
a = l/abs(1-e*e)
c = a*e
if eType == "Hyperbola":
  focus2Dot.set_data([2*c], [0])
  drxX2 = 2*c-(l/e) if e != 0 else 1e6
else: # ellipse
  focus2Dot.set_data([-2*c], [0])
  drxX2 = -2*c-(l/e) if e != 0 else 1e6

focus2Dot.set_visible(True)
drx2.set_data([drxX2, drxX2], [-10, 10])
drx2.set_visible(True)
```

`set_visible()` controls the visibility of the second focus and directrix, which aren't needed when a circle or parabola are being drawn.

conicsPolar.py draws a predetermined ellipse, parabola, or hyperbola with the polar equation $r = l/(1 + e\cos(\theta))$, as shown in Fig. 16.20. Only one branch of the hyperbola is drawn, and only one focus is displayed for the ellipse and hyperbola.

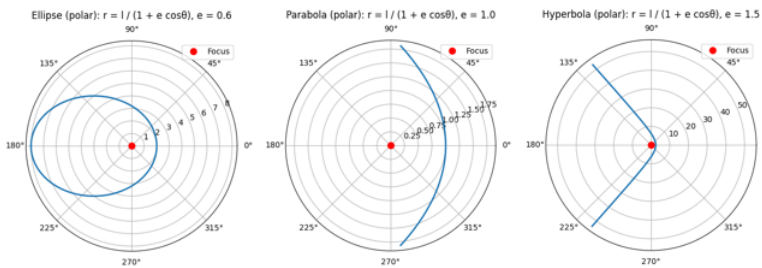


Figure 16.20. `conicsPolar.py`: ellipse, parabola, or hyperbola

The main purpose of this program is to show off matplotlib's polar plotting capabilities since `conicsEccen.py` converts polar coordinates to Cartesian values before they are graphed.

User input of 'e', 'p', or 'h' leads to a three-way branch that draws one of the graphs. The hyperbola code is:

```
elif choice == 'h':
  e = 1.5
  a = 5
  thetaLimit = math.acos(-1/e) - 0.1
  thetaRange = (-thetaLimit, thetaLimit)
  thetas, rs = conicPts(e, a, thetaRange=thetaRange)
  label = "Hyperbola"
```

Note that e and a are fixed before the curve is plotted, while `conicsEccen.py` fixes e and l.

`conicPts()` returns a list of θ's and r's for the polar coordinates of one branch of the hyperbola by using $r = l/(1 + e\cos\theta)$. l is calculated using $l = a(1 - e^2)$, and must deal with $\pm$ values of l depending on the conic type, set a sensible range for the generated angles, and avoid division by zero:

```
def conicPts(e, a, step=0.01, thetaRange=None):
  if e < 1:     # Ellipse
    l = a * (1 - e**2)
  elif abs(e - 1) < 1e-6:  # parabola; v.close to 1
    l = a
  else:         # Hyperbola
    l = a * (e**2 - 1)

  if thetaRange is None:
    thetaStart = -math.pi + 0.0001
    thetaEnd = math.pi - 0.0001
  else:
    thetaStart, thetaEnd = thetaRange

  thetas = []
  rs = []
  theta = thetaStart
  while theta <= thetaEnd:
    denom = 1 + e * math.cos(theta)
    if abs(denom) > 1e-6:  # not close to 0
      r = l / denom
      thetas.append(theta)
      rs.append(r)
    theta += step
  return thetas, rs
```

16.8 Dandelin Spheres

What's the connection between the focus-directrix definition of a conic section and its geometric definition in terms of a sliced cone? The first proof linking the

two was devised by the Belgian mathematician G.P. Dandelin in 1822. It requires a lemma from solid geometry, Oglivy's Wastebasket Theorem: "Two planes parallel to the base of a right circular cone cut off equal segments on all the rulings" [**Ogi68**]. The rulings can be thought of as the slants of the wastebasket in Fig. 16.21.

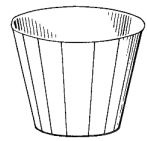


Figure 16.21. The Wastebasket Theorem Illustrated

16.8.1 The Ellipse. The Dandelin proof for an ellipse employs Fig. 16.22.

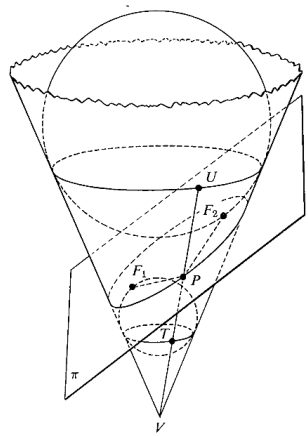


Figure 16.22. Dandelin Spheres and the Ellipse

Let π be a plane that cuts the cone to form an oval. A sphere will fit tightly inside the cone, tangent to it and also to π, touching it at F_1. A second sphere is tangent to π from above, touching it at F_2. F_1 and F_2 will turn out to be the foci of the ellipse.

Choose any point P on the oval. Draw PF_1 and PF_2, which lie in π and are therefore tangent to the lower and upper spheres respectively. Also draw the line $VTPU$ on the surface of the cone, making it tangent to both spheres. Now because tangents to a sphere from an external point P are equal, we have

$$PF_1 = PT \quad \text{and} \quad PF_2 = PU$$

Adding,

$$PF_1 + PF_2 = PT + PU = TU.$$

Now suppose that P is moved to a new position on the oval. TU will take a new position on another line, but by the Wastebasket Theorem, its length will not change. This means that for all positions of P,

$$PF_1 + PF_2 = \texttt{a constant},$$

the definition of an ellipse with foci at F_1 and F_2.

16.8.2 The Ellipse's Eccentricity. Fig. 16.23 shows another ellipse with a focus F and one of its Dandelin spheres.

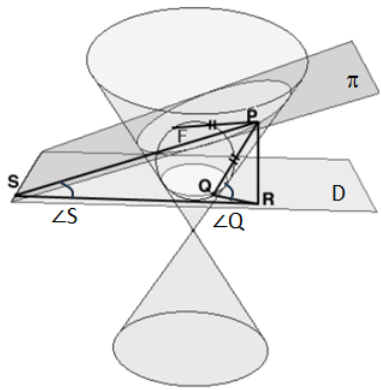


Figure 16.23. Conis Section Angles and the Ellipse

Given P, let Q be the corresponding point on the Dandelin circle; that is, let Q be the point where the segment joining P to the cone apex meets the Dandelin plane D. Let R be the foot of the perpendicular dropped from P to the Dandelin plane, and let S be the foot of the perpendicular dropped from P to the directrix along the plane π.

Since segments PF and PQ are both tangents to the Dandelin sphere, we have $PF = PQ$. But this time we manipulate the focus-directrix ratio for P like so:

$$\frac{PF}{PS} = \frac{PQ}{PS} = \frac{PR/\sin\angle Q}{PS} = \frac{PR}{PS}\frac{1}{\sin\angle Q} = \frac{\sin\angle S}{\sin\angle Q}$$

$\angle S$ is constant as P moves about the curve because it's the angle between the cutting plane π and the Dandelin plane D. But $\angle Q$ is also constant since it's the angle that the surface of the cone makes with D. Therefore, the focus-directrix ratio is also a constant, which again proves that the curve is an ellipse.

We want to go a step further and extract an expression for the eccentricity e in terms of these angles. For now, we'll assume that the conic section is an ellipse (which makes $\angle S$ smaller than $\angle Q$).

It helps to examine the figure side-on (see Fig. 16.24), reducing all of the elements to their intersections with the plane through the cone's axis, perpendicular to the directrix.

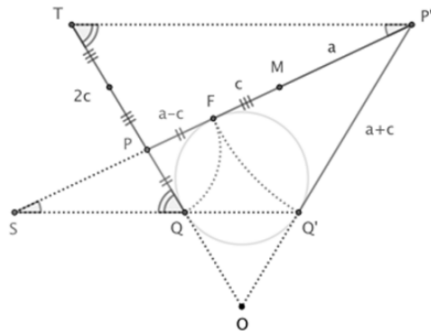


Figure 16.24. Conic Section Angles and the Ellipse from the Side

P is the point on the conic closest to the directrix (which has been projected onto the point S), and P' the farthest point. (Q and Q' are the corresponding points on the Dandelin plane, which has been projected into a line.) Then PP' is the major axis of the ellipse. The ellipse's focus, F, corresponds to the point where the incircle of $\triangle OPP'$ meets PP'; the ellipse's center corresponds to M, the midpoint of PP'.

Now that we know where everything is, a few more applications of the equal-tangent-segment property is all we need. With $a = MP'$ and $c = MF$, we have

$$\frac{\sin\angle S}{\sin\angle Q} = \frac{PT}{PP'} = \frac{QT - QP}{2a} = \frac{(a+c)-(a-c)}{2a} = \frac{2c}{2a} = \frac{c}{a} = e$$

Thus, we've linked a geometric property of the conic section (the ratio of two angles) to the eccentricity of the curve.

16.8.3 The Parabola. For the parabola, the Dandelin Sphere diagram (Fig. 16.25) only employs a single sphere and focus.

Let π be the cutting plane, this time parallel to the rulings on the cone, and let α be the plane determined by the circle of tangency of the sphere. Choose any point P on the (alleged) parabola, and draw PD perpendicular to CD, the line of intersection of α and π. We need another plane, not shown, through P and parallel to α, which cuts the cone in the indicated circle. Let RS be part of

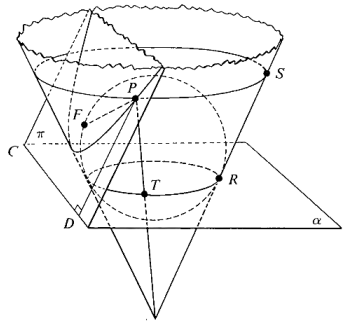


Figure 16.25. Dandelin Spheres and the Parabola

the ruling parallel to π. Then $PD = RS$, because parallel planes cut off equal segments from parallel lines. But $RS = PT$ by the Wastebasket Theorem, and $PT = PF$ because they are tangents to the sphere from P. So we have $PF = PD$, the definition of a parabola; and in addition, the directrix CD is the intersection of α and π.

16.8.4 The Hyperbola. This time we need two spheres, with the second located in the other nappe of the double cone. Since the intersecting plane is inclined near to the axis of the double cone, it meets both branches (see Fig. 16.26).

Consider the two inscribed spheres that touch both the cone and the intersecting plane. The spheres occupy different branches of the cone but lie on the same side of the plane, and so we have

$$\begin{aligned} BF_1 &= BP_1, \quad BF_2 = BP_2, \\ BF_2 - BF_1 &= BP_2 - BP_1 = P_1P_2 = \texttt{a constant}. \end{aligned}$$

16.9 Quadratic Equations

The graph of the quadratic equation

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0,$$

where A, B, and C are not all zero, is nearly always a conic section. In general, A, B, and C define the conic's shape, type, and orientation. More specifically, A controls its curvature in the x-direction, C controls the curvature in the y-direction, and B specifies the curve's rotation. If $B = 0$, then the conic section's axes are aligned with the x and y axes, but if $B \neq 0$, then it'll be rotated by an angle θ such that $\tan 2\theta = \frac{B}{A-C}$.

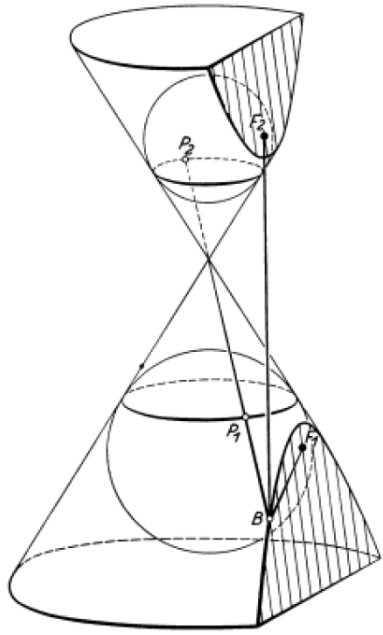


Figure 16.26. Dandelin Spheres and the Hyperbola

The shape/type is determined by the discriminant $\Delta = B^2 - 4AC$:

$$\begin{array}{lll} \Delta < 0 & \Rightarrow & \texttt{ellipse (or circle if A=C)} \\ \Delta = 0 & \Rightarrow & \texttt{parabola} \\ \Delta > 0 & \Rightarrow & \texttt{hyperbola} \end{array}$$

The D coefficient in the quadratic equation shifts the conic section horizontally, while E moves it vertically. This means that its possible to determine the conic's center (for a ellipse or hyperbola) by solving:

$$\frac{\partial}{\partial x}(Ax^2 + Bxy + Cy^2 + Dx + Ey + F) = 0$$
$$\frac{\partial}{\partial y}(Ax^2 + Bxy + Cy^2 + Dx + Ey + F) = 0$$

which reduces to two cases:

$$2Ax + By + D = 0, \quad Bx + 2Cy + E = 0$$

F controls the conic section's scaling. For an ellipse, a larger $|F|$ reduces its size, while it changes how far the branches of a hyperbola are from the origin.

Any conic section can be converted to an axes-aligned form by rotating it to eliminate Bxy, and translating it to eliminate the Dx and Ey terms. The result is a quadratic involving only $A'x^2$, $B'y^2$, and a constant.

16.9.1 Linking the Discriminant and Eccentricity. The discriminant and eccentricity for different conic sections are summarized in Table 16.1.

Conic Section	Discriminant Δ	Eccentricity e
Ellipse (includes circle)	$B^2 - 4AC < 0$	$0 < e < 1$
Parabola	$B^2 - 4AC = 0$	$e = 1$
Hyperbola	$B^2 - 4AC > 0$	$e > 1$

Table 16.1. Conic Discriminant and Eccentricity

$B^2 - 4AC$ and e can be related through the rotated and shifted forms of the quadratic equation.

After an ellipse has been rotated to remove the xy-term (making $B = 0$):

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

with $a > b$, and its eccentricity is:

$$e = \frac{c}{a} = \sqrt{\frac{c^2}{a^2}} = \sqrt{1 - \frac{b^2}{a^2}} < 1.$$

In the quadratic, A and $C > 0 \Rightarrow B^2 - 4AC < 0$.

For a left-facing, axis-aligned parabola $y^2 = 4ax$, $e = 1$, and the quadratic has $B = 0$ and one of A or $C = 0 \Rightarrow B^2 - 4AC = 0$.

The unrotated hyperbola (i.e. $B = 0$):

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

has a similar eccentricity as the ellipse:

$$e = \sqrt{1 + \frac{b^2}{a^2}} > 1.$$

In the quadratic, A and C have opposite signs so $\Rightarrow B^2 - 4AC > 0$.

16.9.2 Rotating a Conic Section. When we rotate a graph's axes by an angle θ, the new coordinates (x', y') can be defined in terms of (x, y):

$$\begin{cases} x = x' \cos\theta - y' \sin\theta \\ y = x' \sin\theta + y' \cos\theta \end{cases}$$

We want to find coefficients A', B', C' so the rotated conic has the same geometric shape as before, but expressed using (x', y'):

$$A'x'^2 + B'x'y' + C'y'^2 + D'x' + E'y' + F' = 0$$

Let's substitute the x and y equations into $Ax^2 + Bxy + Cy^2$:

$$A(x'\cos\theta - y'\sin\theta)^2 + B(x'\cos\theta - y'\sin\theta)(x'\sin\theta + y'\cos\theta) + C(x'\sin\theta + y'\cos\theta)^2$$

Expand all the terms, and collect the powers of x'^2, $x'y'$, and y'^2. After (some considerable) simplification, the new coefficients are:

$$A' = A\cos^2\theta + B\cos\theta\sin\theta + C\sin^2\theta,$$
$$B' = 2(C - A)\sin\theta\cos\theta + B(\cos^2\theta - \sin^2\theta),$$
$$C' = A\sin^2\theta - B\sin\theta\cos\theta + C\cos^2\theta.$$

If the original conic section didn't have a xy term (i.e. $B = 0$), then these simplify to:

$$A' = A\cos^2\theta + C\sin^2\theta,$$
$$B' = 2(A - C)\sin\theta\cos\theta,$$
$$C' = A\sin^2\theta + C\cos^2\theta.$$

Consider an ellipse:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad \Rightarrow \quad A = \frac{1}{a^2},\ C = \frac{1}{b^2},\ B = 0$$

After this ellipse is rotated by an angle θ, we get:

$$\begin{aligned} A' &= (\cos(\theta)^2/a^2) + (\sin(\theta)^2/b^2) \\ B' &= 2 * \sin(\theta) * \cos(\theta) * (1/a^2 - 1/b^2) \\ C' &= (\sin(\theta)^2/a^2) + (\cos(\theta)^2/b^2) \end{aligned}$$

16.9.3 Drawing with the Quadratic Equation. conicsQuad.py gives the user a choice of plotting one of six quadratic equations representing ellipses, parabolas, and hyperbolas:

```
1. Ellipse: x^2/a^2 + y^2/b^2 = 1
2. Rotated Ellipse (30 deg)
3. Parabola: y^2 = 4ax
4. Rotated Parabola (45 deg)
5. Hyperbola: x^2/a^2 - y^2/b^2 = 1
6. Rotated Hyperbola (30 deg)
```

Fig. 16.27 shows what's plotted for each choice.

The graphs use matplotlib's contour plotting, the standard approach when an implicit equation of the form $F(x_1, \ldots, x_n) = 0$ needs to be drawn.

The code evaluates $Ax^2 + Bxy + Cy^2 + Dx + Ey + F$ at 800×800 = 640,000 (x, y) points, creating a grid of results. The contour plot is created by:

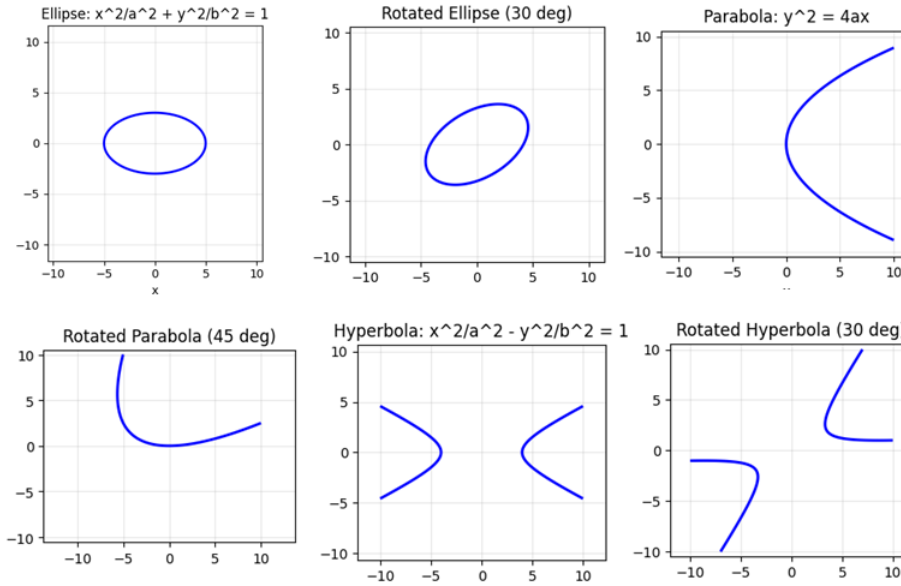


Figure 16.27. Possible `conicsQuad.py` Plots

```
contours = plt.contour(xs, ys, zs, levels=[0],
                                colors='blue', linewidths=2)
```

`xs` and `ys` are lists of the x- and y- coordinates, and `zs` is the grid of results. The `levels` argument specifies that only the (x, y) coordinates with the result $z = 0$ will be plotted. Also, the contour algorithm doesn't just connect the coordinates where the equation equals zero, but also employs linear interpolation between adjacent (x, y) points to approximate where the zero-crossing occurs. For example, if one grid point has a z value 0.5 and its neighbor is -0.3, then the algorithm calculates that the zero must occur approximately 62.5% of the way between them $(0.5/(0.5+0.3))$. Interpolation means that the resulting contour can pass through positions that weren't supplied as (x, y) values, and the result is a smooth continuous curve

Each of the six choices assumes values for a and b, and uses them to calculate A, B, C, D, E, and F for the quadratic. For example, choice 1 (the axis-aligned ellipse) uses:

```
a, b = 5, 3
A = 1 / a**2
B = 0
C = 1 / b**2
```

```
  D = E = 0
  F = -1
```

These are passed to `plotConic()` which generates (x, y) points between -10 and 10 on both axes (coded as two lists of x- and y- values). These are passed to `conicEquation()` to generate z values:

```
def conicEquation(x, y, A, B, C, D, E, F):
  return A*x**2 + B*x*y + C*y**2 + D*x + E*y + F

# : code not shown
zs = []
for y in ys:
  row = []
  for x in xs:
    row.append( conicEquation(x, y, A, B, C, D, E, F))
  zs.append(row)
```

For a rotated conic, a hard-wired `angle` is used along with a and b to determine the quadratic's coefficients. For example, to rotate an ellipse by 30°:

```
  a, b = 5, 3
  angle = math.radians(30)
  A = (math.cos(angle)**2 / a**2) + (math.sin(angle)**2 / b**2)
  B = 2 * math.sin(angle) * math.cos(angle) * (1 / a**2 - 1 / b**2)
  C = (math.sin(angle)**2 / a**2) + (math.cos(angle)**2 / b**2)
  D = E = 0
  F = -1
```

Note that the math used here corresponds to what was worked out in the previous section.

16.10 Parameterized Conic Sections

Single-valued functions, such as $y = x^2$ are easily plotted by plugging a series of x values into the equation, and collecting the output. Unfortunately, multi-valued functions, such as those for the ellipse and the hyperbola, can be more problematic. One approach is to express the x- and y- coordinates as parameterized functions of time t, $x = f(t)$ and $y = g(t)$. Of course, this is particularly useful when studying motion (such as elliptical orbits) because a point's position is determined by time. However, t doesn't need to be time-related, it can just as readily denote an angle or a distance traveled along a curve.

A parameterized equation requires a parameter interval, such as $a \leq t \leq b$, to set the range of generated coordinates – starting at $(f(a), g(a))$, to $(f(b), g(b))$.

16.10.1 The circle $x^2 + y^2 = 1$. The equations and parameter interval

$$x = \cos t, \quad y = \sin t, \quad 0 \leq t \leq 2\pi,$$

describe the position $P(x, y)$ of a point that moves counterclockwise around the circle $x^2 + y^2 = 1$ as t increases (Fig. 16.28).

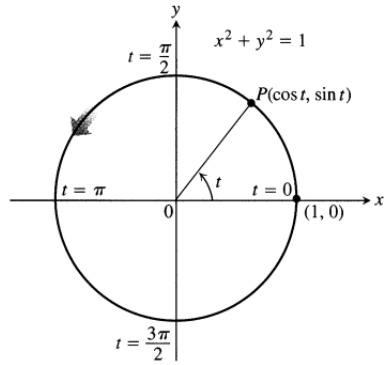


Figure 16.28. A Parameterized Circle

We know the point lies on this circle for every value of t because

$$x^2 + y^2 = \cos^2 t + \sin^2 t = 1.$$

The point starts at $(1, 0)$, moves up and to the left as t approaches $\pi/2$, and continues around the circle to stop back at $(1, 0)$ when $t = 2\pi$.

16.10.2 A Parabola. The points $P(x, y)$ forming a parabola are given by the equations and parameter interval

$$x = t, \quad y = t^2, \quad -\infty < t < \infty.$$

If we eliminate t between the equations, then $y = (t)^2 = x^2$.

As t increases from $-\infty$ to ∞, the point moves down the left arm of the curve, passes through the origin, and up the right-hand side (Fig. 16.29).

16.10.3 The ellipse, $x^2/a^2 + y^2/b^2 = 1$. The points forming an ellipse are given by

$$x = a\cos t, \quad y = b\sin t, \quad 0 \le t \le 2\pi.$$

The Cartesian equation is obtained by eliminating t from:

$$\cos t = \frac{x}{a}, \quad \sin t = \frac{y}{b}$$

with the help of the identity $cos^2 t + sin^2 t = 1$:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1, \quad \text{or} \quad \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

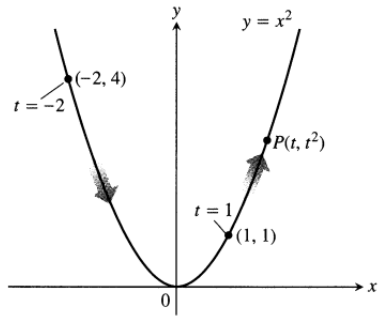


Figure 16.29. A Parameterized Parabola

When $t = 0$, the coordinates are

$$x = a\cos(0) = a, \quad y = b\sin(0) = 0,$$

so the point starts at $(a, 0)$. As t increases, the point moves counterclockwise. It traverses the ellipse once, returning to its starting position $(a, 0)$ at time $t = 2\pi$ (Fig. 16.30).

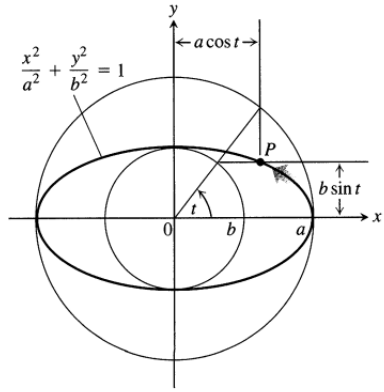


Figure 16.30. A Parameterized Ellipse

16.10.4 The Right-hand Branch of the Hyperbola, $x^2/a^2 - y^2/b^2 = 1$.

$P(x, y)$ at time t is given by

$$x = a\sec t, \quad y = b\tan t, \quad -\frac{\pi}{2} < t < \frac{\pi}{2}$$

The Cartesian equation is obtained by eliminating t, with the help of the identity $\sec^2 t - \tan^2 t = 1$, to yield $x^2/a^2 - y^2/b^2 = 1$.

As t ranges from $-\pi/2$ and $\pi/2$, $x = a \sec t$ remains positive and $y = b \tan t$ runs between $-\infty$ and ∞, so P traverses the hyperbola's right-hand branch. It comes in along the branch's lower half as $t \to 0$, reaches $(a, 0)$ at $t = 0$, and moves into the first quadrant as t increases towards $\pi/2$ (Fig. 16.31).

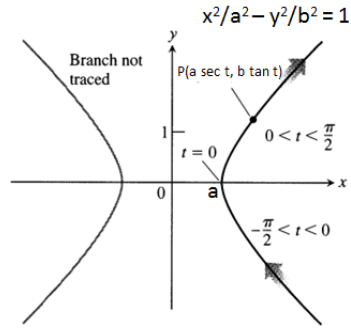


Figure 16.31. Half of a Parameterized Hyperbola

$\pi/2 < t < 3\pi/2$ represents the portion of the hyperbola in the second and third quadrants, but note that $t = \pi/2$ and $t = -\pi/2$ have to be dealt with carefully since the points are at $\pm\infty$.

There are alternative ways to parameterize a hyperbola. For example, the right-hand branch can be expressed as

$$x = a\sqrt{t^2 + 1}, \quad y = bt.$$

Or we can use hyperbolic trigonometric functions:

$$x = \pm a \cosh t, \quad y = b \sinh t.$$

We met this version back in Section 16.3.5, except that because the hyperbola was rotated so its arms turned up and down rather than to the left and right, the equations were $x = a \sinh \theta$ and $y = b \cosh \theta$.

16.11 Drawing Parameterized Equations

We've a general-purpose plotting program for parameterized equations in paramPlot.py and paramUtils.py.

`paramPlot.py` reads parameters supplied in a text file. For example:

```
python paramPlot.pt parabola.txt
```

parabola.txt contains:

```
title = parabola
x = t
y = a*t**2
tMin = -5
tMax = 5
step = 0.1
```

The parameterized equation are the two lines $x = ...$ and $y = ...$, and the interval for t is set with `tMin`, `tMax`, and `step`, while `title` is used to label the plot. These values can be omitted, and `paramPlot.py` will switch to defaults.

The plot (see Fig. 16.32) includes two sliders for `a` and `b` variables which may appear in the parameterized equations. We only use `a` in the parabolic equation given above.

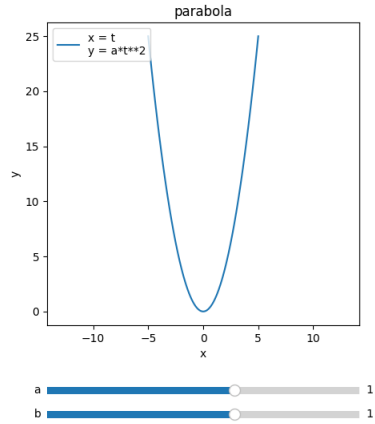


Figure 16.32. Plot of a Parameterized Parabola

The parameter files for the ellipse and hyperbola are ellipse.txt and hyperbola.txt. Their key lines are:

```
# ellipse.txt
a = 3
x = a * math.cos(t)
y = b * math.sin(t)
```

and

```
# hyperbola.txt
a = 3
x = a * math.cosh(t)
y = b * math.sinh(t)
```

No b is specified, which causes the plotter to use $b = 1$. The graphs are shown in Fig. 16.33.

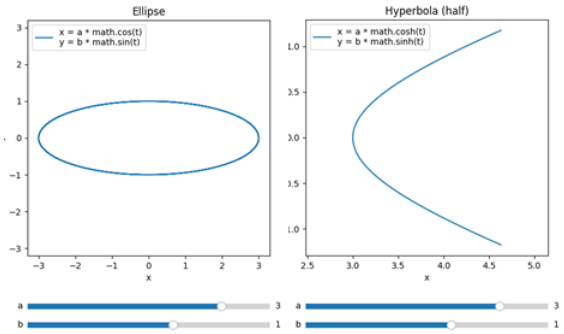


Figure 16.33. Plots of a Parameterized Ellipse and Hyperbola

16.12 Reflections in Conic Sections

The reflection problem was first considered by Heron (c.150 BC to 250 AD) in his *Catoptrica* text concerned with mirrors and the behavior of light. What position for a point P on a line L minimizes the sum of the distances $F_1P + PF_2$ between two fixed points F_1 and F_2 on the same side of the line (see Fig. 16.34)?

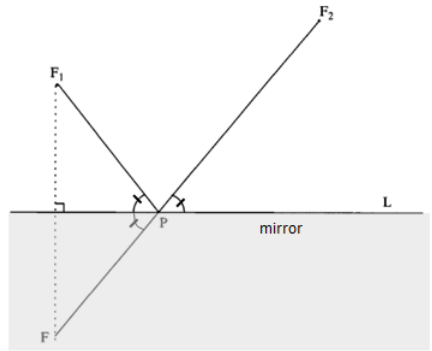


Figure 16.34. The Heron Reflection Problem

Reflect F_1P in L to create FP of the same length. The length of F_1PF_2 equals FPF_2 and so we can choose P to minimize the former by minimizing the latter.

We select P so F, P, and F_2 lie in a line, which means that P is located where FF_2 meets L.

The angles that F_1P and F_2P make with L are equal, which Heron used to infer the Law of Reflection: the angle of an incident ray equals the angle of reflection since light is naturally efficient and takes the shortest path possible via a reflecting line. Therefore, a ray travels from one point to another in the least possible time, now known as Fermat's Principle of Least Time.

16.12.1 Parabolic Reflection. We'll explain how reflection applies to a parabola by referring to Fig. 16.35. In the figure, $PF = PD$ by definition. Draw DF, bisect it at M, and connect PM. Then, $\triangle FMP$ and $\triangle DMP$ have three equal sides, and so are congruent, making $\angle 1 = \angle 2$. But $\angle 2 = \angle 3$ (they're vertical angles), so $\angle 1 = \angle 3$.

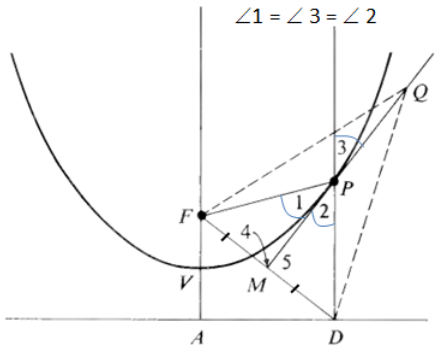


Figure 16.35. The Parabolic Reflection Problem

It remains to be shown that PM, which so far is just a line connecting two points, is actually a tangent to the parabola at P. Let's assume that it isn't, and so intersects the curve for a second time at some point Q. $\triangle FQM$ is congruent to $\triangle DQM$, because they share QM, $FM = MD$, and $\angle 4 = \angle 5$. These are corresponding parts of *thecongruent* $\triangle FMP$ and $\triangle DMP$, and so $FQ = QD$, making QD the perpendicular distance to AD by applying the definition of a parabola. In other words, Q and P must coincide, making PM a tangent at P.

This means that light entering a parabolic mirror parallel to its axis of symmetry (in Fig. 16.35 this is a ray travelling straight down to hit P) will reflect into its focus, F.

Reflecting telescopes use parabolic mirrors because they gather a great deal of light into one spot (the focus) making objects visible that otherwise would be too dim to see. Parabolic mirrors are also used in solar collectors, long distance

microphones, and receiving antennas. Going in 'reverse', light radiating out from the focus reflects to become rays parallel to the axis of symmetry, making parabolic reflectors ideal for headlights, spotlights, and directional transmitting antennas. A common feature of most science museums are two large concave parabolic dishes, serving as acoustic mirrors, facing each other in a room designated as a whispering gallery (see Fig. 16.36).

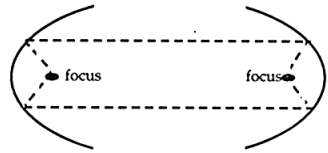


Figure 16.36. Parabolic Mirrors in a Whispering Gallery

16.12.2 Elliptic Reflection. The reflection property for an ellipse is similar to the parabolic case, but causes light reflected from one focus (F_1) to travel to the other focus (F_2) (see Fig. 16.37).

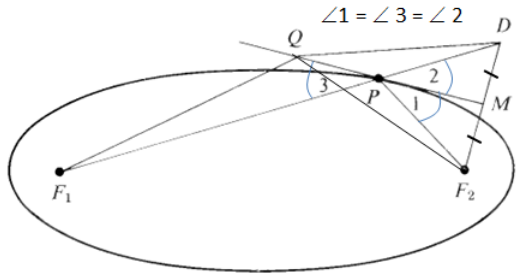


Figure 16.37. The Elliptical Reflection Problem

Focus-to-focus mirroring can be proved by extending F_1P in the figure so $PD = PF_2$. Draw DF_2, bisect it at M, and connect PM. The result, as in the parabola's case, is that $\angle 1 = \angle 3$.

To show that PM is a tangent we again assume a second point Q where PM cuts the curve (thereby implying that it isn't a tangent). $QD = QF_2$ as before, but if Q is on the curve then

$$F_1Q + QD = F_1Q + QF_2 = F_1P + PF_2 = F_1P + PD.$$

The first and last members of this equality state

$$F_1Q + QD = F_1D$$

which is impossible unless Q coincides with P.

16.12.3 Hyperbolic Reflection. The reflection property for the hyperbola is closely related to elliptic reflection, but the focus-to-focus link is less obvious. In Fig. 16.38, a beam emitted from the focal point F_2 is reflected to the right in such a way that $\angle 1 = \angle 3 = \angle 2$. If the reflected ray is extended backwards it forms a line that passes through the other focal point F_1.

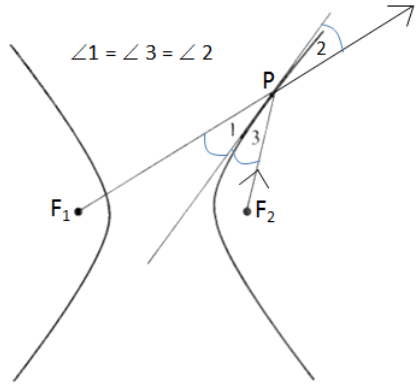


Figure 16.38. The Hyperbolic Reflection Problem

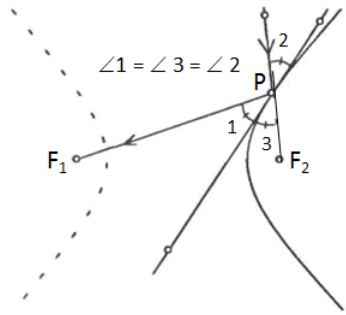


Figure 16.39. Another Hyperbolic Reflector

Fig. 16.39 shows a variant where light is projected onto the right-hand mirror from the left. The reflected ray is directed towards F_1, but if the original beam is extended, it'll pass through F_2. The physical effect is different, but it relies on the

same underlying equality as Fig. 16.38, that $\angle 1 = \angle 3 = \angle 2$. The proof is very similar to that for the elliptic reflector, so will be skipped.

Some telescopes use a hyperbolic mirror in addition to a main parabolic reflector to redirect the light from the main focus to a more convenient point (see Fig. 16.40). The parabola and hyperbola share the same focus, F_2. Light hitting the parabolic mirror reflects toward F_2, bounces off the hyperbolic mirror, and travels to its other focus at F_1. It's much easier to place a telescope eyepiece at this point rather than at F_2.

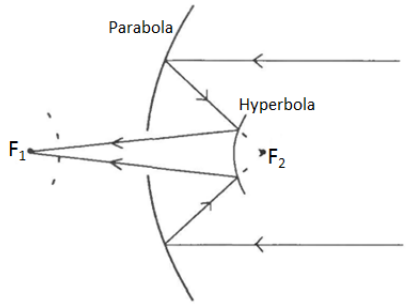


Figure 16.40. Compound Parabolic-Hyperbolic Reflector

16.12.4 Drawing Conic Reflections. `paraRays.py` and `ellipseRays.py` implement reflection by treating light rays as vectors, using our `Vec.py` class (see section 14.4 on quaternions).

Fig. 16.41 gives an overview of the problem: a vector $\mathbf{v}$ is reflected off the plane p around the normal vector $\mathbf{n}$. The goal is to define the reflection $\mathbf{w}$ in terms of $\mathbf{v}$.

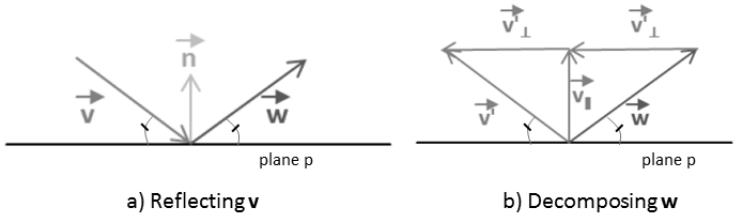


Figure 16.41. Reflection as Vectors

We start by splitting $\mathbf{v}$ into its parallel component $\mathbf{v}_{\parallel}$ and perpendicular component $\mathbf{v}_{\perp}$. Fig. 16.41(b) shows the reflection again, but with $\mathbf{v}$ negated as $\mathbf{v}'$. If

we follow the path from the start of $\mathbf{w}$ along $\mathbf{v}'$ and take two steps along $\mathbf{v}'_\perp$, we reach the head of $\mathbf{w}$. In other words:

$$\mathbf{w} = \mathbf{v}' - 2(\mathbf{v}'_\perp).$$

As $\mathbf{v}' = -\mathbf{v}$, we can rewrite $\mathbf{w}$ in terms of $\mathbf{v}$ after some work involving the dot product (see section 10.3):

$$\begin{aligned}
\mathbf{w} &= -\mathbf{v} - 2(-\mathbf{v}_\perp) \\
&= -\mathbf{v} - 2(-\mathbf{v} - (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}) \\
&= -\mathbf{v} + 2(\mathbf{v} + (-\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}) \\
&= -\mathbf{v} + 2\mathbf{v} + 2(-\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} \\
&= \mathbf{v} + 2(-\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} \\
&= \mathbf{v} - 2(\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}
\end{aligned}$$

Both `paraRays.py` and `ellipseRays.py` use vector code very like this. For example:

```
w = v - (2 * v.dot(n) * n)
```

However, $\mathbf{n}$, the normal vector, is more complex than the one in Fig. 16.41 (where $\mathbf{n} = \mathbf{v}_\parallel$) since the reflecting surface is curved.

16.12.5 The Parabolic Reflector. `paraRays.py` generates several random vectors heading downwards to hit an upward facing parabola. The reflections are calculated, and all of them pass through the parabola's focus (see Fig. 16.42).

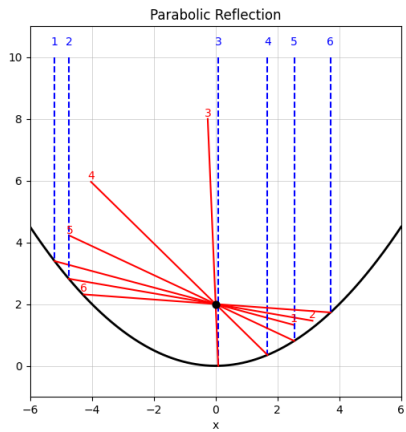


Figure 16.42. Parabolic Reflection

The following code excerpt shows how a single downward facing vector reflects off the parabola.

```
def parabolaY(x, f):
  return (x * x) / (4.0 * f)

def reflectVector(v, n):
  return v - n * (2 * v.dot(n))


downVec = Vec(0, -1, 0)  # downward vector

$ x is a random x-coordinate
yHit = parabolaY(x, f)
pHit = Vec(x, yHit, 0)  # contact with parabola

# convert parabola slope at x into a normal
m = x / (2.0 * f)
normal = Vec(-m, 1, 0).normalize()

w = reflectVector(downVec, normal)
```

The code hardwires the parabola as $y = x^2/(4f)$, which means that its slope at any x-coordinate is $m = x/(2f)$. The slope of its normal will be $-1/m$ which is converted into a vector by using $-m$ and 1 as the normal's (x, y) coordinates.

16.12.6 The Elliptic Reflector. ellipseRays.py generates several vectors leaving the left focus of an ellipse. The reflected vectors are calculated, and the plot shows that they all pass through the ellipse's other focus (see Fig. 16.43).

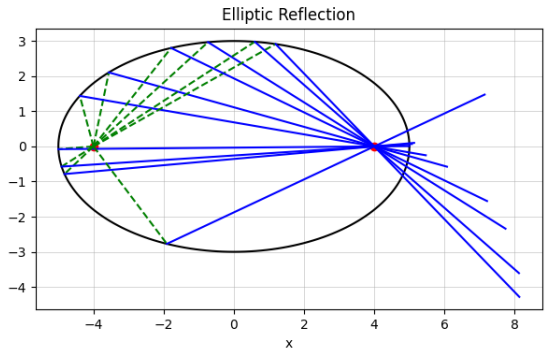


Figure 16.43. Elliptic Reflection

One tricky aspect of this program is obtaining the coordinates where the rays emitted from the left-hand focus intersect the ellipse. The second problem is calculating the normal at an intersection so the reflection can be generated.

16.12.7 Intersecting a Ray with the Ellipse. A ray is parameterized as:

$$x = x_0 + t \cdot d_x, \quad y = y_0 + t \cdot d_y$$

where (x_0, y_0) is the origin of the ray, (d_x, d_y) is its direction, and $t \geq 0$. These equations are substituted into the ellipse equation:

$$(x_0 + t \cdot d_x)^2/a^2 + (y_0 + t \cdot d_y)^2/b^2 = 1$$

This is expanded and rearranged into a quadratic of the form:

$$At^2 + Bt + C = 0$$

where:

$$\begin{aligned} A &= (d_x^2/a^2) + (d_y^2/b^2) \\ B &= 2(x_0 \cdot d_x/a^2 + y_0 \cdot d_y/b^2) \\ C &= (x_0^2/a^2) + (y_0^2/b^2) - 1 \end{aligned}$$

We solve this quadratic and use its positive root to calculate the intersection. Here's the code:

```
def intersectPt(origin, dir):
  x0, y0 = origin.x, origin.y
  dx, dy = dir.x, dir.y

  A = (dx*dx)/(a*a) + (dy*dy)/(b*b)
  B = 2*x0*dx/(a*a) + 2*y0*dy/(b*b)
  C = (x0*x0)/(a*a) + (y0*y0)/(b*b) - 1

  # get positive intersection distance
  t = (-B + math.sqrt(B*B - 4*A*C)) / (2*A)
  return origin + dir*t
```

16.12.8 Calculating a Normal to the Ellipse. For an ellipse centered at the origin:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

differentiate both sides with respect to x:

$$\frac{2x}{a^2} + \frac{2y}{b^2}\frac{dy}{dx} = 0.$$

Simplify:

$$\frac{dy}{dx} = -\frac{b^2 x}{a^2 y}.$$

This gives the slope of the tangent at any point (x, y). The normal is perpendicular to the tangent, so its slope is the negative reciprocal:

$$m = \frac{a^2 y}{b^2 x}.$$

A vector pointing in the direction of the normal is proportional to

$$\mathbf{n} = (1, m) = (1, \frac{a^2 y}{b^2 x}).$$

Since any scalar multiple of a point leaves the direction unchanged, we can write

$$\mathbf{n} = (\frac{x}{a^2}, \frac{y}{b^2}).$$

This is used to calculate the reflection $\mathbf{r}$ of the ray $\mathbf{d}$:

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

Here's the function:

```
def reflectRay(hitPt, d):
  nx = hitPt.x /(a*a)
  ny = hitPt.y /(b*b)
  n = Vec(nx, ny, 0).normalize()
  return d - n * (2 * d.dot(n))
```

16.13 Confocal Conic Sections

Two conic sections are confocal if they share the same focus. Fig. 16.44 shows three confocals 'families' involving parabolas, ellipses, and hyperbolas.

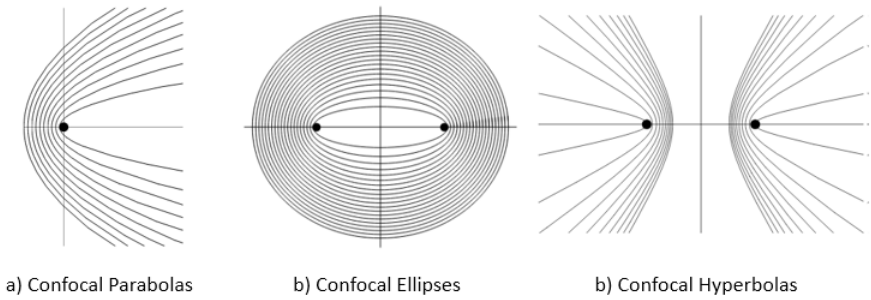


Figure 16.44. Confocal Families

A visually pleasing example is when confocal hyperbolas and ellipses are brought together, forming an orthogonal net of curves (see Fig. 16.45)

It's quite straightforward to prove that these curves always intersect at 90° if we utilize the reflection properties for the ellipse and hyperbola from the last section, combined as in Fig. 16.46 [**Ogi90**].

Referencing the angles marked on Fig. 16.46, we know

$$\angle 1 = \angle 3 \quad \texttt{(ellipse reflection)}$$
$$\angle 2 = \angle 4 \quad \texttt{(hyperbola reflection)}$$

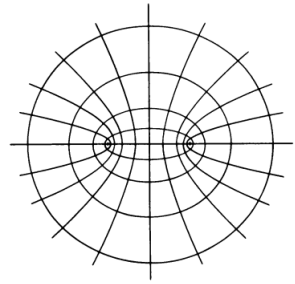


Figure 16.45. Confocal Ellipses and Hyperbolas

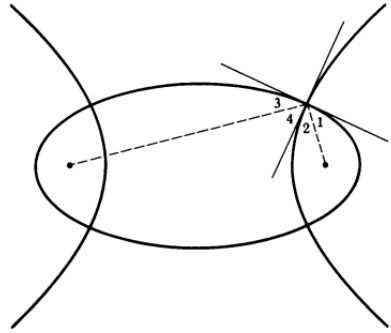


Figure 16.46. Confocal Ellipses and Hyperbolas are Orthogonal

Adding,

$$\angle 1 + \angle 2 = \angle 3 + \angle 4$$

Since we also know that $\angle 1 + \angle 2 + \angle 3 + \angle 4$ is a straight line (the ellipse's tangent), we can conclude that

$$\angle 1 + \angle 2 = \angle 3 + \angle 4 = 90°$$

Another visually interesting example are confocal parabolas opening in opposite directions, as in Fig. 16.47.

It is similarly easy to prove that these curves always intersect at 90° by utilizing the parabola's reflection properties as in Fig. 16.48.

Referencing the angles marked on the figure, the reflection property, applied to parabola α, means that:

$$\angle 1 = \angle 2$$

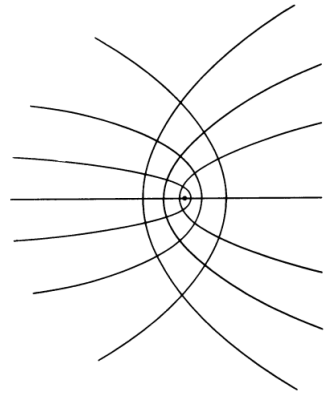


Figure 16.47. Confocal Opposite-facing Parabolas

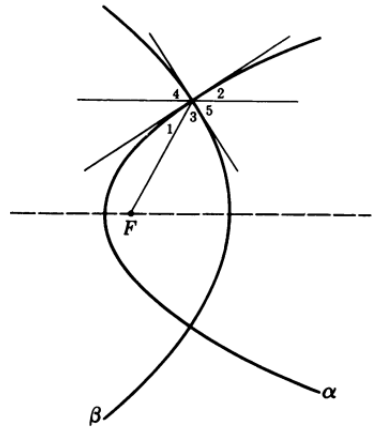


Figure 16.48. Confocal Opposite-facing Parabolas are Orthogonal

and with β as the reflector,

$$\angle 3 = \angle 4 = \angle 5.$$

Adding equals to equals,

$$\angle 1 + \angle 3 = \angle 2 + \angle 5$$

Once again the total of these four angles is a straight line, so each sum is a right angle.

16.13.1 Drawing Confocal Conics. confocalEl.py draws a single ellipse and multiple hyperbolas all with the same foci $(\pm c, 0)$. It's 'obvious' by looking at Fig. 16.49 that all of the hyperbolas intersect with the ellipse at 90°.

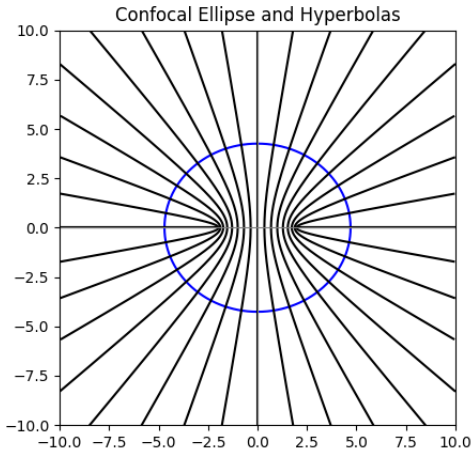


Figure 16.49. Confocal Ellipse and Hyperbolas

`confocalEl.py` utilizes elliptical coordinates ((μ, ν) and (`mu`, `nu`) in the code), which lend themselves to solving problems involving elliptical symmetry (the sums of distances from two foci). Aside from the obvious cases of ellipses and hyperbola, many real-world problems have this property, such as:

- Orbital mechanics;
- Acoustics (ellipse-shaped whispering galleries);
- Antenna radiation with two foci;
- Potential from two charges (dipoles).

(μ, ν) are utilized in the parameterization:

$$x = c \cosh \mu \cos \nu \qquad y = c \sinh \mu \sin \nu$$

where $c > 0$ is the semi-focal distance, $0 \leq \mu < \infty$, and $0 \leq \nu < 2\pi$. Note that μ and ν have different ranges. μ is used in cosh() and sinh(), hyperbolic functions that grow, while ν is employed in cos() and sin() which are periodic.

Curves of constant μ form ellipses with foci at $(\pm c, 0)$:

$$\frac{x^2}{(c \cosh \mu)^2} + \frac{y^2}{(c \sinh \mu)^2} = 1$$

By assuming a fixed $\mu = \mu_0$ in $x = c\cosh\mu\cos\nu$ and $y = c\sinh\mu\sin\nu$, then

$$\frac{x}{c\cosh\mu_0} = \cos\nu, \qquad \frac{y}{c\sinh\mu_0} = \sin\nu.$$

Square and add:

$$\left(\frac{x}{c\cosh\mu_0}\right)^2 + \left(\frac{y}{c\sinh\mu_0}\right)^2 = \cos^2\nu + \sin^2\nu = 1.$$

This is the equation for an ellipse with a semi-major axis $a = c\cosh\mu_0$, and semi-minor axis $b = c\sinh\mu_0$. The focal distance is

$$f = \sqrt{a^2 - b^2} = \sqrt{c^2\cosh^2\mu_0 - c^2\sinh^2\mu_0} = \sqrt{c^2} = c$$

so the foci are at $(\pm c, 0)$.

Curves of constant ν form hyperbolas with the same foci:

$$\frac{x^2}{(c\cos\nu)^2} - \frac{y^2}{(c\sin\nu)^2} = 1$$

By assuming a fixed $\nu = \nu_0$ in $x = c\cosh\mu\cos\nu$ and $y = c\sinh\mu\sin\nu$, then

$$\frac{x}{c\cos\nu_0} = \cosh\mu, \qquad \frac{y}{c\sin\nu_0} = \sinh\mu.$$

Square and subtract:

$$\left(\frac{x}{c\cos\nu_0}\right)^2 - \left(\frac{y}{c\sin\nu_0}\right)^2 = \cosh^2\mu - \sinh^2\mu = 1.$$

This is the equation for a hyperbola with $a = c\cos\nu_0$, and $b = c\sin\nu_0$. The focal distance is:

$$f = \sqrt{a^2 + b^2} = \sqrt{c^2(\cos^2\nu_0 + \sin^2\nu_0)} = c$$

so the foci are again at $(\pm c, 0)$.

The relevant parts of the code:

```
def ellipsePlot(c, mu):  # Fixed mu
  xs = []; ys = []
  for i in range(361):
    nu = 2 * math.pi * i / 360   # vary nu to get coords
    x = c * math.cosh(mu) * math.cos(nu)
    y = c * math.sinh(mu) * math.sin(nu)
    xs.append(x); ys.append(y)
  return xs, ys

def drawHyperbola(c, nu): # Fixed nu
  xs = []; ys = []
  for k in range(1, 800):
    mu = 0.01 + k * 0.01   # vary mu to get coords
    x = c * math.cosh(mu) * math.cos(nu)
    y = c * math.sinh(mu) * math.sin(nu)
```

```
    if abs(x) > 10:  # stop if x goes beyond plot range
      break
    xs.append(x); ys.append(y)
  return xs, ys
```

confocalPara.py draws confocal parabolas with the same focus $(f, 0)$ but different directrixes, $x = d$. When $d < f$, the parabolas open to the right, and with $d > f$ they open to the left. Once again, it's 'obvious' from looking at Fig. 16.50 that the opposite-facing parabolas all intersect at 90°.

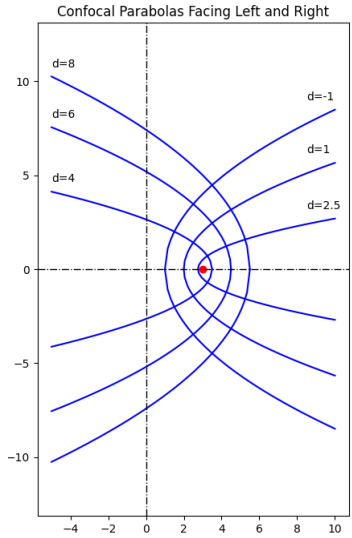


Figure 16.50. Confocal Opposite-facing Parabolas

The parabola's coordinates are generated by a call to `parabolaPlot(xMin, xMax, f, d)` with different values for `d`:

```
f = 3       # focus at (3,0)
ds = [-1, 1, 2.5, 4, 6, 8] # directrix positions

for d in ds:
  xs, ys = parabolaPlot(-5, 10, f, d)
  plt.plot(xs, ys, 'b')
  plt.plot(xs, [-y for y in ys], 'b')
```

The function only returns a list of coordinates for the part of the parabola above the x-axis, which explains the second call to `plt.plot()` to draw the lower half.

The formulation of the parabola in terms of just its focal part $(f, 0)$ and directrix $x = d$ means that any point (x, y) on the parabola satisfies

$$\sqrt{(x-f)^2 + y^2} = |x - d|.$$

Squaring both sides gives

$$(x-f)^2 + y^2 = (x-d)^2.$$

Expanding and simplifying,

$$x^2 - 2fx + f^2 + y^2 = x^2 - 2dx + d^2,$$
$$y^2 = 2(f-d)x + (d^2 - f^2).$$

Let $p = (f-d)/2$ which implies that

$$f + d = 2(p + d), \quad f - p = p + d$$

Also set

$$x_v = f - p = f - \left(\frac{f-d}{2}\right) = \frac{f+d}{2}.$$

x_v is the x-coordinate of the point located at the shortest distance between the focus and directrix.

Then the equation becomes

$$\begin{aligned} y^2 &= 2(f-d)x + (d-f)(d+f) \\ y^2 &= 2 \cdot 2px - 2p \cdot 2(p+d) \\ y^2 &= 4px - 4p(f-p) \\ y^2 &= 4p(x - x_v) \end{aligned}$$

which is the standard form for a parabola opening left or right. This equation style is reflected in `parabolaPlot()`:

```
def parabolaPlot(xMin, xMax, f, d):
  p = (f - d) / 2
  vX = f - p
  if xMax == xMin:
    return [], []
  stepSize = (xMax - xMin) / steps
  xsRange = [xMin + i * stepSize for i in range(steps+1)]
  # Insert vX in sorted order if within range;
  # avoids gaps in the plotted points
  if xMin <= vX <= xMax:
    bisect.insort(xsRange, vX)

  xs =  []; ys = []
  for x in xsRange:
    ySqu = 4 * p * (x - vX)
    if ySqu >= 0:  # only store positive y's
      ys.append(ySqu ** 0.5)
```

```
        xs.append(x)
    return xs, ys
```

One messy aspect is the need to ensure that x_v (vX in the code) is included in the list of plotted coordinates. It may be missed due to the way that the plotting is discretized by stepSize, and because negative y-values are dropped from the coordinates list.

Exercises

(1) The circular waves in Fig. 16.51 were made by touching the surface of a ripple tank, first at A and then at B. As the waves expand, their points of intersection appear to trace a hyperbola. Does they really do that? To find out, we can model the waves with circles centered at A and B.

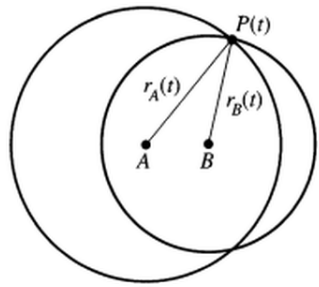


Figure 16.51. Circular Waves

At time t, the point P is $r_A(t)$ units from A and $r_B(t)$ units from B. Since the radii of the circles increase at a constant rate, the speed of the waves is

$$\frac{dr_A}{dt} = \frac{dr_B}{dt}.$$

Work out from this equation that $r_A - r_B$ has a constant value, so that P must lie on a hyperbola with foci at A and B.

(2) *The Trammel of Archimedes.* The mechanical system in Fig. 16.52 consists of a rigid bar of length L, with one end attached to a roller that moves along the y-axis. At a fixed distance R from that end, the bar is attached to a second roller on the x-axis. Let P be the point at the free end of the bar and let θ be the angle the bar makes with the positive x-axis.

a) Find parametric equations for the path of P in terms of θ.

b) Find an equation in x and y whose graph is the path of P, and identify this path.

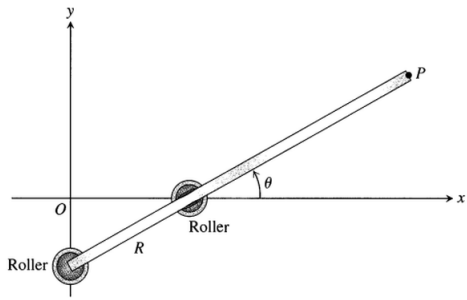


Figure 16.52. The Trammel of Archimedes

(3) *Perihelion and Aphelion.* A planet travels about its sun in an ellipse whose semi-major axis has length a as depicted in Fig. 16.53.

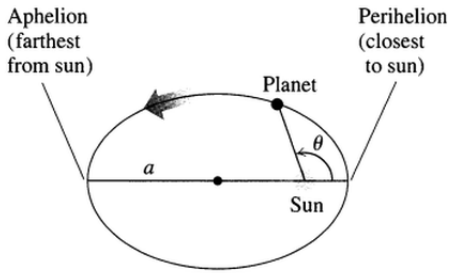


Figure 16.53. A Planet Orbiting a Sun

a) Show that $r = a(1 - e)$ when the planet is closest to the sun and $r = a(1 + e)$ when the planet is farthest away.
b) Use the data in Table 16.2 to determine how close each planet comes to the sun and how far away.

(4) *A satellite orbit.* A satellite passes over the North and South Poles. When it's over the South Pole, it's at the highest point of its orbit, 1000 miles above the surface. Above the North Pole it's at the lowest point, 300 miles.

a) Assuming that the orbit is an ellipse with one focus at the center of the Earth, find its eccentricity. (Take the diameter of the Earth to be 8000 miles.)
b) Using the north-south axis of the Earth as the x-axis and the center of the Earth as the origin, find a polar equation for the orbit.

Planet	Semi-major Axis (AU)	Eccentricity
Mercury	0.3871	0.2056
Venus	0.7233	0.0068
Earth	1.000	0.0167
Mars	1.524	0.0934
Jupiter	5.203	0.0484
Saturn	9.539	0.0543
Uranus	19.18	0.0460
Neptune	30.06	0.0082
Pluto	39.44	0.2481

Table 16.2. Planets Axes and Eccentricities

(5) A comet moves in a parabolic orbit with the sun at the focus. When the comet is 4×10^7 miles from the sun, the line from the comet to the sun makes a 60° angle with the orbit's axis, as shown in Fig. 16.54. How close will the comet come to the sun?

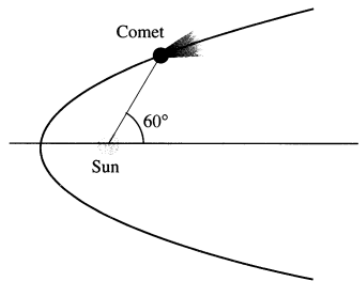


Figure 16.54. A Comet Orbiting the Sun

(6) Perhaps the best mathematical treatment of turtle graphics is *Turtle Geometry: The Computer as a Medium for Exploring Mathematics* by Abelson and Disessa [**AD86**]. The Python version of their simple function for drawing a circle is:

```
def circle(t):
  for _ in range(360):
    t.forward(1)
    t.right(1)
```

while the following functions:

```
def duopoly(t, side1, angle1, side2, angle2):
  c = 0
  for _ in range(200):
    vector(t, c*angle1, side1)
    vector(t, c*angle2, side2)
    c += 1

def vector(t, direction, length):
  t.setheading(direction)
  t.forward(length)
```

are capable of drawing a wide range of patterns, but the specific call:

```
duopoly(t, 4, 2, 8, -2)
```

draws an ellipse. You can try out these functions for yourself in ellipsesTG.py.

a) Represent the (x, y) position of the turtle in terms of parameterized equations involving `length` and `direction`. Then consider how (x, y) changes over time.

b) Use a similar (x, y) parameterization for `duopoly()` and see how the specific call affects (x, y).

Answers

(1)

$$\frac{dr_A}{dt} = \frac{dr_B}{dt} \Rightarrow \frac{d}{dt}(r_A - r_B) = 0 \Rightarrow r_A - r_B = C,$$

So the points $P(t)$ lie on a hyperbola with foci at A and B.

(2) (a) Let C be the point where the vertical line through P meets the x-axis. We have $AB = R$ and $AP = L \Rightarrow PB = L - R$. Then $y = PB\sin\theta \Rightarrow y = (L - R)\sin\theta$, and $BC = PB\cos\theta \Rightarrow BC = (L - R)\cos\theta$. Also, in $\triangle OAB$, $\cos\theta = \frac{OB}{AB} = \frac{OB}{R} \Rightarrow OB = R\cos\theta$. Therefore, $x = OB + BC = R\cos\theta + (L - R)\cos\theta = L\cos\theta \Rightarrow x = L\cos\theta$ and $y = (L - R)\sin\theta$.

(b) Note that

$$\frac{x^2}{L^2} + \frac{y^2}{(L-R)^2} = \frac{L^2\cos^2\theta}{L^2} + \frac{(L-R)^2\sin^2\theta}{(L-R)^2} = 1.$$

Therefore, the points $P(x, y)$ satisfy the equation

$$\frac{x^2}{L^2} + \frac{y^2}{(L-R)^2} = 1,$$

which is an ellipse.

Planet	Perihelion (AU)	Aphelion (AU)
Mercury	0.3075	0.4667
Venus	0.7184	0.7282
Earth	0.9833	1.0167
Mars	1.3817	1.6663
Jupiter	4.9512	5.4548
Saturn	9.0210	10.0570
Uranus	18.2977	20.0623
Neptune	29.8135	30.3065
Pluto	29.6549	49.2251

Table 16.3. Planet Perihelion & Aphelion

(3) (a) Perihelion $= a - ae = a(1-e)$, Aphelion $= ea + a = a(1+e)$

(b) See Table 16.3.

(4) (a) The length of the major axis is 300 miles + 8000 miles + 1000 miles $= 2a$ $\Rightarrow a = 4650$ miles. If the center of the Earth is one focus and the distance from the center to the satellite's low point is 4300 miles (half the diameter plus the distance above the North Pole), then the distance from the center of the ellipse to the focus (center of the Earth) is 4650 miles - 4300 miles = 350 miles $= c$. Therefore

$$e = \frac{c}{a} = \frac{350 \text{ miles}}{4650 \text{ miles}} = \frac{7}{93}.$$

(b) $r = \frac{a(1-e^2)}{1+e\cos\theta} \Rightarrow r = \frac{4650[1-(7/93)^2]}{(1+(7/93)\cos\theta)} = \frac{430000}{93+7\cos\theta}$ miles.

(5) If the vertex is $(0,0)$, then the focus is $(p,0)$. Let $P(x,y)$ be the present position of the comet. Then

$$\sqrt{(x-p)^2+y^2} = 4\times 10^7.$$

Since $y^2 = 4px$ we have

$$\sqrt{(x-p)^2+4px} = 4\times 10^7 \Rightarrow (x-p)^2+4px = 16\times 10^{14}.$$

Also, $x - p = 4\times 10^7 \cos 60^\circ = 2\times 10^7 \Rightarrow x = p + 2\times 10^7$. Therefore

$$\begin{aligned}
(2\times 10^7)^2 + 4p(p+2\times 10^7) &= 16\times 10^{14}\\
4\times 10^{14} + 4p^2 + 8p\times 10^7 &= 16\times 10^{14}\\
4p^2 + 8p\times 10^7 - 12\times 10^{14} &= 0\\
p^2 + 2p\times 10^7 - 3\times 10^{14} &= 0\\
(p+3\times 10^7)(p-10^7) &= 0\\
p = -3\times 10^7 \quad &\text{or} \quad p = 10^7.
\end{aligned}$$

Since p is positive we obtain $p = 10^7$ miles.

(6) (a) The circle's parametric form

$$x(t) = r\cos t, \quad y(t) = r\sin t, \quad t \in [0, 2\pi]$$

In this case, $r = 1$ and $t = 1°$.

Over 360 iterations it will create a circle with a circumference of 360 'turtle' units. Thus the circle's radius will be $360/2\pi \approx 57$ units.

(b) The parametric form for an ellipse:

$$x(t) = a\cos(t), \quad y(t) = b\sin(t)$$

When `duopoly(4, 2, 8, -2)` is called, the turtle draws two vectors:

- vector($2c$, 4): heading $2c$, length 4
- vector($-2c$, 8): heading $-2c$, length 8

In Cartesian coordinates, the position after multiple iterations is:

$$x(c) \approx 4 \cdot \cos(2c) + 8 \cdot \cos(-2c)$$
$$y(c) \approx 4 \cdot \sin(2c) + 8 \cdot \sin(-2c)$$

Since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$:

$$\begin{aligned} x(c) &= 4 \cdot cos(2c) + 8 \cdot cos(2c) = 12 \cdot cos(2c) \\ y(c) &= 4 \cdot sin(2c) - 8 \cdot sin(2c) = -4 \cdot sin(2c) \end{aligned}$$

This is an ellipse with parameter $t = 2c$:

$$x = 12 \cdot \cos(t), \quad y = -4 \cdot \sin(t)$$

This gives us an ellipse with:

- Semi-major axis $a = 12$ (horizontal)
- Semi-minor axis $b = 4$ (vertical)

In summary: when angle1 = -angle2, the two vectors create a sum and difference pattern: the sum ($4+8 = 12$) determines one axis, while the absolute difference ($|4 - 8| = 4$) determines the other.