Growth

Produced by Bruce Carpenter Published by Math Everywhere, Inc.
www.matheverywhere.com

1.09 Parametric Plotting Basics

B.1) Parametric plots in two dimensions: Circular parameters

A handy way to plot the circle

$x^2 + y^2 = 9$

is to write

x[t] = 3 Cos[t]

and

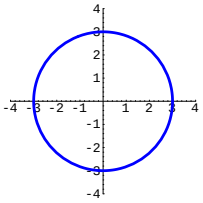
y[t] = 3 Sin[t]

and then to plot the points

{x[t], y[t]} = {3 Cos[t], 3 Sin[t]}

as t advances from 0 to 2π.

```
Clear[x, y, t];
{x[t_], y[t_]} = {3 Cos[t], 3 Sin[t]};

circle =
  ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
   PlotStyle -> {{Blue, Thickness[0.015]}},
   AspectRatio -> Automatic,
   PlotRange -> {{-4, 4}, {-4, 4}}];
```



In this way, the original variables x and y are plotted by means of a third variable (t in this case) called a parameter. A parameter is an auxiliary variable that plays a back-room role.

□B.1.a)

When you plot the circle of radius 3 centered at {0, 0} parametrically through the parametric formula

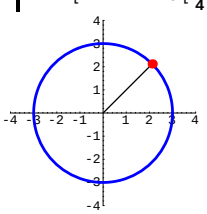
{x[t], y[t]} = {3 Cos[t], 3 Sin[t]},

what is the physical meaning of the parameter t?

□Answer:

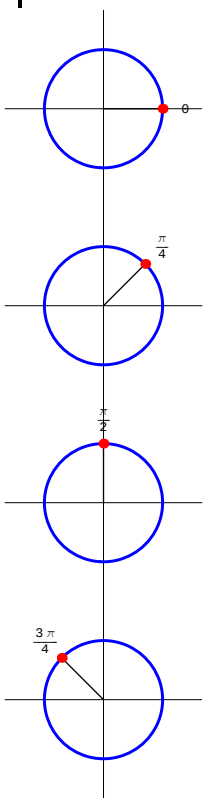
Look at this plot showing the circle and the point {x[t], y[t]} you get with $t = \frac{\pi}{4}$:

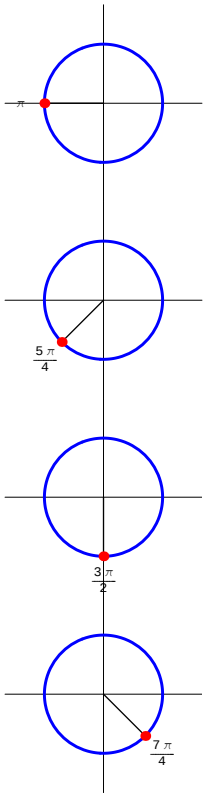
```
Clear[ray, t]
ray[t_] = Graphics[
    {Line[{{0, 0}, {x[t], y[t]}}],
   {Red, PointSize[0.05], Point[{x[t], y[t]}]}}];
Show[circle, ray[π/4]];
```

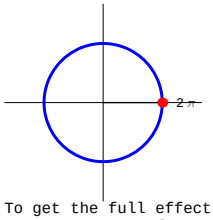


Now look at this:

```
Table[Show[circle, ray[t],
    Graphics[Text[t, 1.4 {x[t], y[t]}]],
    AspectRatio -> Automatic, Ticks -> None,
    PlotRange -> {{-5, 5}, {-5, 5}}],
   {t, 0, 2π, π/4}];
```







To get the full effect, grab all the plots and press command-Y and run forward at a speed of your own choice.

The labels on the graphs give the values of t that make {x[t], y[t]} plot out at the indicated point.

□B.1.b)

Parameters sometimes give you plotting freedom normal plotting does not allow. Some curves are best described via parametric equations and are more difficult to describe in the usual y = f[x] terms.
To see what this means, plot the spiral

x = x[t] = t Cos[t]
y = y[t] = t Sin[t].

□Answer:

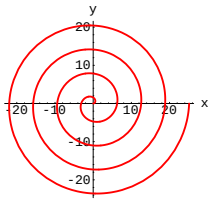
```
Clear[x, y, t]
x[t_] = t Cos[t]
```

t Cos[t]

```
y[t_] = t Sin[t]
```

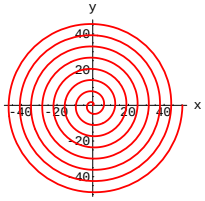
t Sin[t]

```
ParametricPlot[{x[t], y[t]}, {t, 0, 8 π},
  AxesLabel -> {"x", "y"},
  AspectRatio -> Automatic,
  PlotStyle -> {{Thickness[0.01], Red}}];
```



Extend the values of the parameter t to see more of the spiral:

```
ParametricPlot[{x[t], y[t]}, {t, 0, 16 π},
  AxesLabel -> {"x", "y"},
  AspectRatio -> Automatic,
  PlotStyle -> {{Thickness[0.01], Red}}];
```

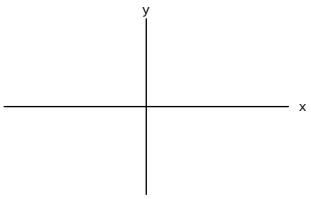


Bull's eye.

B.2) Parametric plots of curves in three dimensions

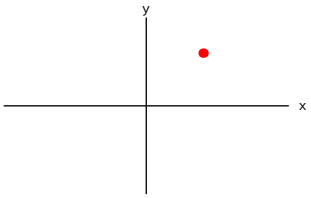
When you plot in two dimensions, you use two coordinate axes:

```
h = 5;
spacer = h/10;
twodims =
  Graphics[{
    Line[{{-h, 0}, {h, 0}}],
      Text["x", {h + spacer, 0}],
    Line[{{0, -h}, {0, h}}],
      Text["y", {0, h + spacer}]}];

Show[twodims, PlotRange -> All, AspectRatio -> 1/GoldenRatio];
```



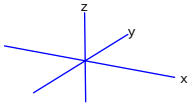
To plot a point, you specify an {x, y} coordinate:

```
{x, y} = {2, 3};
Show[twodims,
  Graphics[{Red, PointSize[0.03], Point[{x, y}]}],
  PlotRange -> All];
```



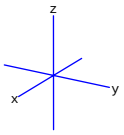
To plot in three dimensions, you need three coordinate axes:

```
h = 5;
spacer = h/10;
threedims =
  Graphics3D[{
    {Blue, Line[{{-h, 0, 0}, {h, 0, 0}}]},
      Text["x", {h + spacer, 0, 0}],
    {Blue, Line[{{0, -h, 0}, {0, h, 0}}]},
      Text["y", {0, h + spacer, 0}],
    {Blue, Line[{{0, 0, -h}, {0, 0, h}}]},
      Text["z", {0, 0, h + spacer}]}];

Show[threedims, PlotRange -> All, Boxed -> False,
 AspectRatio -> 1/GoldenRatio];
```



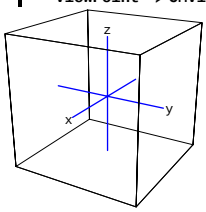
This is *Mathematica*'s default view. Usually Calculus&*Mathematica* uses the viewpoint CMView ={2.7, 1.6, 1.2}.
Here is how the three axes look from this viewpoint:

```
CMView = {2.7, 1.6, 1.2};
Show[threedims, PlotRange → All, ViewPoint -> CMView, Boxed -> False];
```



Add a framing box:

```
Show[threedims, PlotRange -> All,
  ViewPoint -> CMView];
```



□B.2.a)

What is the advantage of using the CMView?

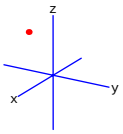
□Answer:

This way the x – axis points to the left, the y – axis points to the right and the z – axis points up. It's the orientation we usually see in other math and science texts.

□B.2.b)

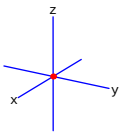
To plot a point in three dimensions, you specify an {x, y, z} coordinate:

```
{x, y, z} = {-1, -3, 3};
CMView = {2.7, 1.6, 1.2};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```

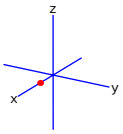


To see what's going on, play with these:

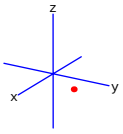
```
{x, y, z} = {0, 0, 0};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```



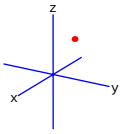

```
{x, y, z} = {2, 0, 0};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```



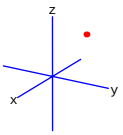

```
{x, y, z} = {2, 3, 0};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```




```
{x, y, z} = {2, 3, 4};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```



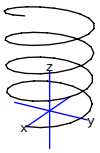

```
{x, y, z} = {0, 3, 4};
Show[threedims,
  Graphics3D[{Red, PointSize[0.03], Point[{x, y, z}]}],
  PlotRange -> All, ViewPoint -> CMView, Boxed -> False];
```



Play with some more of your own choice.

A three-dimensional curve related to a circle is a screw or helix. The x and y coordinates plot out to a circle, but the z coordinate lifts the curve higher and higher:

```
Clear[x, y, z, t]
radius = 5;

x[t_] = radius Cos[t];

y[t_] = radius Sin[t];

z[t_] = t/2;

screw =
 ParametricPlot3D[{x[t], y[t], z[t]},
   {t, 0, 8 π}, DisplayFunction -> Identity];
   Show[threedims, screw,
   PlotRange -> All, ViewPoint -> CMView, Boxed -> False,
   DisplayFunction -> $DisplayFunction, Axes -> None];
```



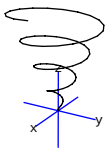
Plot a three-dimensional curve related to a spiral.

□Answer:

The tornado.

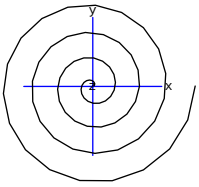
To get a tornado, you use the spiral in the x and y slots and use the z slot to lift the curve:

```
Clear[x, y, z, radius, t]
radius[t_] = t/4;

x[t_] = radius[t] Cos[t];

y[t_] = radius[t] Sin[t];

z[t_] = t/2;

tornado =
 ParametricPlot3D[{x[t], y[t], z[t]},
   {t, 0, 8 π}, DisplayFunction -> Identity];

Show[threedims, tornado,
   PlotRange -> All, ViewPoint -> CMView,
   Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



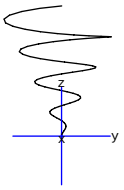
You can look down from above along the z – axis:

```
Show[threedims, tornado,
   PlotRange -> All, ViewPoint -> {0, 0, 4},
   Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



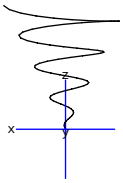
Or from a side along the x – axis:

```
Show[threedims, tornado,
   PlotRange -> All, ViewPoint -> {4, 0, 0},
   Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



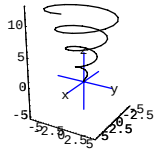
Or from the other side along the y – axis:

```
Show[threedims, tornado,
   PlotRange -> All, ViewPoint -> {0, 4, 0},
   Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



If you want to see actual numbers on the axes, include the option Axes → True:

```
Show[threedims, tornado,
   PlotRange -> All, ViewPoint -> CMView,
   Boxed -> False,
   DisplayFunction -> $DisplayFunction, Axes -> True];
```



□B.2.c)

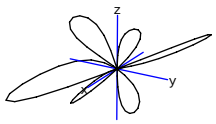
Do something crazy.

□Answer:

Okay.

Take the code for the tornado and put in some new functions for x, y, and z:

```
Clear[x, y, z, radiusx, radiusy, t]
radiusx[t_] =  t Sin[t] ;
radiusy[t_] =  t;

x[t_] = radiusx[t] Cos[t];

y[t_] = radiusy[t] Sin[t];

z[t_] = 8 Sin[t]/2;

crazy =
 ParametricPlot3D[{x[t], y[t], z[t]},
   {t, -2 π, 4 π}, DisplayFunction -> Identity];

Show[threedims, crazy,
   PlotRange -> All, ViewPoint -> CMView,
   Boxed -> False,
   DisplayFunction -> $DisplayFunction, Axes -> None];
```



You can probably do something quite a bit crazier than this. Try it.

B.3) Parametric plots of surfaces in three dimensions

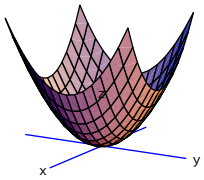
Here is a plot of the surface $z = x^2 + y^2$ in three dimensions:

The ThreeAxes command used below is from the intializations of this lesson. Check it out:

```
?ThreeAxes
ThreeAxes[a,b] makes a standard cartesian axis
  graphics object with x, y, and z running
  from -a to a, and with axis labels b units
  beyond the tips of the axes.  ThreeAxes[a]
  is ThreeAxes[a,a/8].
```

Now try it out on a plot of the surface $z = x^2 + y^2$ in 3 D:

```
Clear[x, y, f]
f[x_, y_] = x^2 + y^2;

surface =
Plot3D[f[x, y], {x, -2, 2}, {y, -2, 2},
     DisplayFunction -> Identity];

threedims = ThreeAxes[3];
Show[threedims, surface, ViewPoint -> CMView,
  PlotRange -> All, Boxed -> False,
  AspectRatio -> 1/GoldenRatio,
  DisplayFunction -> $DisplayFunction];
```



This plot consists of all points {x, y, f[x, y]} as x and y run through the square $-2 \le x \le 2$ and $-2 \le y \le 2$. The surface plotted is over this square in the xy – plane.

This is not a parametric plot.

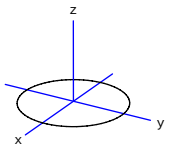
□B.3.a)

Use parametric plotting to plot the same surface over everything inside the circle $x^2 + y^2 = 4$ in the xy – plane.

□Answer:

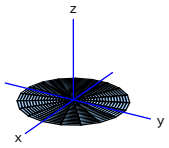
Here is a plot of the circle $x^2 + y^2 = 4$ in the xy – plane:

```
Clear[x, y, t]
x[t_] = 2 Cos[t];
y[t_] = 2 Sin[t];
circle =
ParametricPlot3D[{x[t], y[t], 0},
    {t, 0, 2 Pi},
    DisplayFunction -> Identity];
threedims = ThreeAxes[3];

Show[threedims, circle, ViewPoint -> CMView,
  PlotRange -> All, Boxed -> False,
  AspectRatio -> Automatic,
  DisplayFunction -> $DisplayFunction];
```



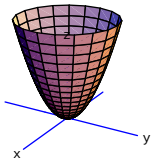
Here is a plot of everything on the xy – plane that is inside this circle:

```
Clear[x, y, r, t]
x[r_, t_] = r Cos[t];
y[r_, t_] = r Sin[t];
disk =
ParametricPlot3D[{x[r, t], y[r, t], 0},
     {r, 0, 2}, {t, 0, 2 Pi},
     DisplayFunction -> Identity];

Show[threedims, disk, ViewPoint -> CMView,
  PlotRange -> All, Boxed -> False,
  DisplayFunction -> $DisplayFunction];
```



Here is a plot of the part of the surface $z = x^2 + y^2$ that is over everything inside the circle $x^2 + y^2 = 4$ in the xy – plane.

```
Clear[x, y, z, r, t, f]
f[x_, y_] = x^2 + y^2;

x[r_, t_] = r Cos[t];
y[r_, t_] = r Sin[t];
z[r_, t_] = f[x[r, t], y[r, t]];

surface =
ParametricPlot3D[{x[r, t], y[r, t], z[r, t]},
     {r, 0, 2}, {t, 0, 2 π},
     DisplayFunction -> Identity];
threedims = ThreeAxes[3];

Show[threedims, surface, ViewPoint -> CMView,
  PlotRange -> All, Boxed -> False,
  DisplayFunction -> $DisplayFunction];
```



This is a parametric plot. And it's a very nice one.

□B.3.b)

Use parametric plotting to plot the surface

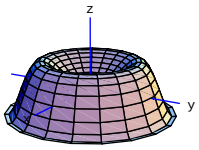
$$z = \text{Sin}[x^2 + y^2]$$

over everything inside the circle $x^2 + y^2 = 5$ in the xy – plane.

□Answer:

Sweat it not.

```
Clear[x, y, z, r, t, f]
f[x_, y_] = Sin[x^2 + y^2];

x[r_, t_] = r Cos[t];
y[r_, t_] = r Sin[t];
z[r_, t_] = f[x[r, t], y[r, t]];

surface =
ParametricPlot3D[{x[r, t], y[r, t], z[r, t]},
     {r, 0, Sqrt[5]}, {t, 0, 2 π},
     DisplayFunction -> Identity];

threedims = ThreeAxes[2.5];

Show[threedims, surface, ViewPoint -> CMView,
  PlotRange -> All, Boxed -> False,
  AspectRatio -> 1/GoldenRatio,
  DisplayFunction -> $DisplayFunction];
```



This is another parametric plot.

Fill it with ice cream and enjoy.

B.4) Derivatives for curves given parametrically: The cycloid

□B.4.a)

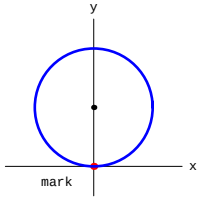
Sometimes parametric formulas are the only viable way of setting up a precise description of a curve. Here is a case:
A circular wheel of radius r rolls along the x – axis. The path traced out by a point P marked on the circle is called a cycloid.
Give parametric formulas {x[t], y[t]} for the cycloid.
Plot in the case that $r = 1$.

□Answer:

Set things up at the start like this:

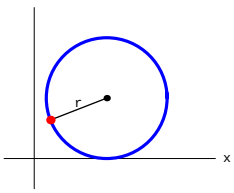
```
Clear[x, y, t];
origin =
  Graphics[{Red, PointSize[0.04],
    Point[{0, 0}]}];
mark = Graphics[Text["mark", {0, 0}, {2, 2}]];
wheel = Graphics[{Thickness[0.015], Blue, Circle[{0, 1}, 1]}];
center = Graphics[{PointSize[0.03], Point[{0, 1}]}];
```

```
Show[origin, wheel, center, mark, Axes -> Automatic,
  AxesOrigin -> {0, 0}, AxesLabel -> {"x", "y"},
  AspectRatio -> Automatic, Ticks -> None,
  PlotRange -> {{-1.5, 1.5}, {-0.5, 2.5}}];
```



After the circle has rolled to the right a bit, the set-up looks like this:

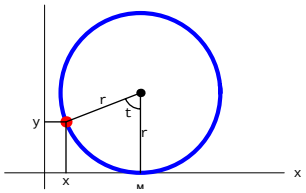
```
mark = Graphics[{Red, PointSize[0.04],
  Point[{1.2 - Sin[1.2], 1 - Cos[1.2]}]}];

wheel = Graphics[{Thickness[0.015], Blue, Circle[{1.2, 1}, 1]}];

center = Graphics[{PointSize[0.03], Point[{1.2, 1}]}];

radiusline =
  Graphics[Line[{{1.2 - Sin[1.2], 1 - Cos[1.2]}, {1.2, 1}}]];
rlabel = Graphics[Text["r",
  {1.2 - Sin[1.2]/2, 1.35 - 1/2 Cos[1/2]}]];

Show[wheel, center,
  radiusline, rlabel, mark,
  Axes -> Automatic,
  AxesOrigin -> {0, 0}, AxesLabel -> {"x", ""},
  AspectRatio -> Automatic, Ticks -> None,
  PlotRange -> {{-.5, 3}, {-.5, 2.5}}];
```



Now put in some labels:

Do not worry about the graphic specifications. When the problem is over, then you'll realize how they were set.

```
yline =
  Graphics[Line[{{0, 1 - Cos[1.2]},
  {1.2 - Sin[1.2], 1 - Cos[1.2]}}]];
ylabel =
Graphics[Text["y", {-.1, 1 - Cos[1.2]}]];
xline =
  Graphics[Line[{{1.2 - Sin[1.2], 0},
      {1.2 - Sin[1.2], 1 - Cos[1.2]}}]];
xlabel = Graphics[Text["x", {1.2 - Sin[1.2], -.1}]];
mlabel = Graphics[Text["M", {1.2, 0}, {0, 2}]];
tline =
  Graphics[Line[{{1.2, 0}, {1.2, 1},
  {1.2 - Sin[1.2], 1 - Cos[1.2]}}]];
tarc =
  Graphics[Circle[{1.2, 1}, .2,
      {3 Pi/2 - 1.2, 3 Pi/2}]];
tlabel = Graphics[Text["t", {1.05, .75}]];

rrlabel = Graphics[Text["r", {1.25, .5}]];

diagram =
 Show[wheel, mark, center, yline, ylabel,
   xline, xlabel, mlabel, tline, tarc, tlabel,
   rlabel, rrlabel,
  Axes -> True, AxesOrigin -> {0, 0},
  AxesLabel -> {"x", ""},
  AspectRatio -> Automatic,
  Ticks -> None,
  PlotRange -> {{-0.5, 3}, {-0.2, 2.1}}];
```



You can find the position {x[t], y[t]} of the mark in terms of the angle measurement t:

Read the parametric formula for y right off the plot:

```
Clear[r, x, y, t]
y[t_] = r - r Cos[t]
```

```
r - r Cos[t]
```

Remember Cos[angle] = adjacent / hypothenuse.

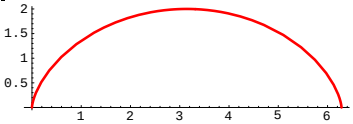
The center is sitting at {r t, r} because the length of the arc on the circle from the point of contact M to the mark is r t. This tells you how to read off the parametric formula for x:

```
x[t_] = r t - r Sin[t]
```

```
r t - r Sin[t]
```

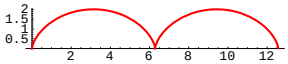
Here comes the plot of the curve traced out by the mark of the first roll in the case that r = 1:

```
r = 1;
firstroll =
ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
 AspectRatio -> Automatic,
 PlotStyle -> {{Red, Thickness[.008]}}];
```



Two rolls:

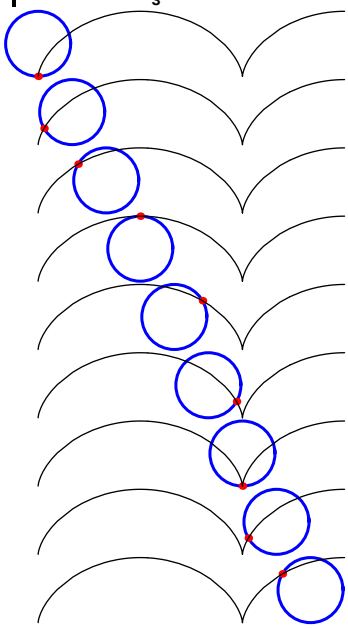
```
ParametricPlot[{x[t], y[t]}, {t, 0, 4 Pi},
  AspectRatio -> Automatic, PlotStyle ->
  {{Red, Thickness[.008]}}];
```

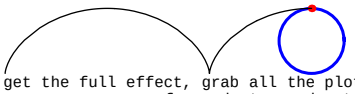


Lots of folks call that corner a cusp.

How about a movie for the case r = 1 and a roll and a half:

```
r = 1;
Clear[mark, wheel, t]
mark[t_] = Graphics[{Red, PointSize[0.02],
           Point[{x[t], y[t]}]}];
wheel[t_] = Graphics[{Thickness[0.008], Blue, Circle[{t, 1}, 1]}];
rollandahalf = ParametricPlot[
  {x[t], y[t]}, {t, 0, 3 Pi}, DisplayFunction -> Identity];

Table[Show[wheel[t], mark[t], rollandahalf,
AxesOrigin -> {0, 0}, AxesLabel -> {"x", ""},
  AspectRatio -> Automatic, Ticks -> None,
  PlotRange -> {{-1.2, 3 Pi + 1}, Automatic},
  DisplayFunction -> $DisplayFunction],
  {t, 0, 3 Pi, Pi/3}];
```





To get the full effect, grab all the plots, press command-Y, and run forward at a moderate speed.

Play with this, and when you're done, send your movie to Siskel and Ebert.

□B.4.b)

Sometimes a curve, like the cycloid above, comes via parametric equations

x = x[t] and y = y[t]

and it is clear that the curve is the graph of a function y = f[x]. This means that y[t] = f[x[t]] for all t's.

Finding the explicit form of f[x] may be impossible or too much trouble.

It might seem that in this situation you are out of luck in calculating the derivative D[f[x], x] = f'[x], but there is a way to do it.

How?

□Answer:

If y = f[x], then clearly

y[t] = f[x[t]].

The chain rule says

y'[t] = f'[x[t]] x'[t].

Consequently

f'[x[t]] = y'[t] / x'[t].

This tells you that to calculate f'[x*] for a given number x*, you find a t* with x[t*] = x* and then you have

f'[x*] = y'[t*] / x'[t*].

□B.4.c)

Here is one roll of the cycloid for a wheel of radius r = 2:

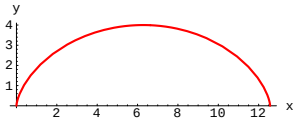
```
Clear[r, x, y, t]
x[t_] = r t - r Sin[t]
```

```
r t - r Sin[t]
```

```
y[t_] = r - r Cos[t]
```

```
r - r Cos[t]
```

```
r = 2;
firstroll =
ParametricPlot[{x[t], y[t]}, {t, 0, 2 π},
 AspectRatio -> Automatic, AxesLabel -> {"x", "y"},
 PlotStyle -> {{Red, Thickness[.008]}}];
```

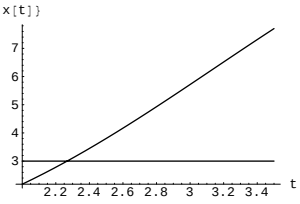


You can see that there is a function f[x] so that this curve is the plot of f[x]; but you (and everyone else) will not be able to find a formula for f[x].
In spite of this, estimate the instantaneous growth rate f'[3].

□Answer:

To get a hold of f'[3], you've got to determine a t∗ that gives x[t∗] = 3:

```
a = 2;
b = 3.5;
Plot[{x[t], 3}, {t, a, b}, AxesOrigin -> {a, x[a]},
 AxesLabel -> {"t", "x[t]"}];
```



t∗ = 2.3 is a reasonable first estimate; get a better estimate:

```
FindRoot[x[t] == 3, {t, 2.3}]
```

```
{t → 2.26717}
```

Try it out:

```
tstar = 2.26717
x[tstar]
```

```
2.26717
2.99999
```

Good enough.

A good estimate of f'[3] is:

```
y'[tstar]
x'[tstar]
```

```
0.467378
```

And you're out of here.

1.09 Parametric Plotting Tutorials

T.1) Parametric plotting for projectile motion

A projectile is fired from a cannon at ground level. The cannon is inclined at an angle a (here $0 < a < \pi/2$) with the horizontal, with a given muzzle speed determined by the explosive charge. If you neglect air resistance, it's not a big deal to come up with parametric formulas for the position {x[t], y[t]} of the projectile t seconds after firing.
Here t measures time in seconds from the instant the cannon is fired.
x[t] measures the horizontal distance in feet of the projectile down range from the cannon at time t.
y[t] measures the height of the projectile in feet at time t.
Gravity acts on the y[t] component of the position but does not act on the x[t] component. Thus

x''[t] = 0 and y''[t] = −32.

The muzzle speed, s, splits into horizontal and vertical components as follows:

x'[0] = s Cos[a]
y'[0] = s Sin[a].

Saying that the cannon is fired at ground level is to say

{x[0], y[0]} == 0.

See what {x[t], y[t]} look like:

```
Clear[x, y, t, a, s, Derivative]
DSolve[{x''[t] == 0,
    x[0] == 0,
    x'[0] == s Cos[a]}, x[t], t]
```

```
{{x[t] → s t Cos[a]}}
```

```
DSolve[{y''[t] == -32,
    y[0] == 0,
    y'[0] == s Sin[a]}, y[t], t]
```

```
{{y[t] → -16 t^2 + s t Sin[a]}}
```

For a given muzzle speed s and cannon angle a:

```
x[t_, s_, a_] = s t Cos[a]
```

```
s t Cos[a]
```

```
y[t_, s_, a_] = -16 t^2 + s t Sin[a]
```

```
-16 t^2 + s t Sin[a]
```

Now you can go to work.

□T.1.a)

Plot the trajectory of the projectile given that the cannon is inclined at an angle $\frac{\pi}{6}$ with the horizontal and with muzzle speed 200 ft/sec.

□Answer:

See when the projectile is back on the ground:

```
Solve[y[t, 200, π/6] == 0, t]
```

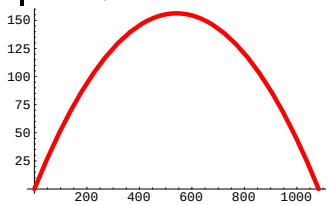
```
{{t → 0}, {t → 25/4}}
```

Here comes the plot:

```
Clear[trajectoryplotter]
trajectoryplotter[t_] = {x[t, 200, π/6], y[t, 200, π/6]}
```

```
{100 √3 t, 100 t - 16 t^2}
```

```
ParametricPlot[trajectoryplotter[t], {t, 0, 25/4},
 AspectRatio → 1/GoldenRatio,
 PlotStyle → {{Red, Thickness[0.015]}}];
```



There it is, a beautiful parabolic arch.

□T.1.b)

Measure the horizontal range of the cannon as a function of muzzle speed and angle of inclination.
For a given muzzle speed, what angle of inclination maximizes the horizontal range of the cannon?

□**Answer:**

Go back to:

```
Clear[x, y, t, s, a, t];
x[t_, s_, a_] = s t Cos[a]
```

```
s t Cos[a]
```

```
y[t_, s_, a_] = -16 t^2 + s t Sin[a]
```

```
-16 t^2 + s t Sin[a]
```

Now find when the projectile hits the ground:

```
Solve[y[t, s, a] == 0, t]
```

$\{\{t \to 0\}, \{t \to \frac{1}{16} s \text{Sin}[a]\}\}$

For a given muzzle speed s and angle a, the horizontal range of the cannon is:

```
x[1/16 s Sin[a], s, a]
```

$\frac{1}{16} s^2 \text{Cos}[a] \text{Sin}[a]$

For a given muzzle speed, this is as big as it can be when the angle a is set so that

Cos[a] Sin[a]

is as big as it can be.

Now examine the derivative of Cos[a] Sin[a] with respect to a:

```
Expand[∂a (Cos[a] Sin[a]), Trig -> True]
```

```
Cos[a]^2 - Sin[a]^2
```

So

D[Cos[a] Sin[a], a] = 0

for

Cos[2 a] = 0;

which gives

$2\,a = \frac{\pi}{2}$;

this is the same as

$a = \frac{\pi}{4}$.

For a given muzzle speed, you get the greatest horizontal range by setting the firing angle a equal to $\frac{\pi}{4}$.

Not much of a surprise.

□T.1.c)

How do you use the parametric formulas:

```
Clear[x, y, t, s, a, t]
x[t_, s_, a_] = s t Cos[a]
```

```
s t Cos[a]
```

```
y[t_, s_, a_] = -16 t^2 + s t Sin[a]
```

```
-16 t^2 + s t Sin[a]
```

to explain why, no matter what the muzzle speed and the angle are, the projectile moves on a parabola?

□**Answer:**

Look at the parametric formulas:

```
Clear[x, y, t, s, a, t]
x[t_, s_, a_] = s t Cos[a]
```

```
s t Cos[a]
```

```
y[t_, s_, a_] = -16 t^2 + s t Sin[a]
```

```
-16 t^2 + s t Sin[a]
```

If you put x = x[t, s, a] and y = y[t, s, a] and then you eliminate t by solving for t in terms of y and substituting the result into the formula for x, you get:

```
ExpandAll[Eliminate[{x == x[t, s, a],
                     y == y[t, s, a]}, t]]
```

```
16 x^2 - s^2 x Cos[a] Sin[a] == -s^2 y Cos[a]^2
```

This tells you that when you fix a muzzle speed s and a firing angle a, then the projectile moves on the curve

$y = -\frac{16\,x^2\,\text{Sec}[a]^2}{s^2} + x\,\text{Tan}[a]$.

For a given muzzle speed and angle, this curve is a parabola because no powers of x other than x^2, x^1, and x^0 are present.

T.2) Parametric plotting for designing a cam

□T.2.a)

A cam is to be built with a smooth shape and such that the following central displacements occur at the given angles with the horizontal:

$30^\circ \to 2.03$
$85^\circ \to 1.75$
$175^\circ \to 2.27$
$270^\circ \to 1.58$
$315^\circ \to 1.38$

To see what this means, feed in the given information:

```
Clear[displacement]
{displacement[1] = 2.03,
 displacement[2] = 1.75,
 displacement[3] = 2.07,
 displacement[4] = 1.58,
 displacement[5] = 1.38};
```

```
This instruction makes use of the Mathematica Package
Miscellaneous`Units` which is included in the initialization cells of
                              this lesson.
```

```
Needs["Miscellaneous`Units`"]
```

```
Clear[angle]
{angle[1] = π Convert[30 Degree, Radian][[1]],
 angle[2] = π Convert[85 Degree, Radian][[1]],
 angle[3] = π Convert[175 Degree, Radian][[1]],
 angle[4] = π Convert[270 Degree, Radian][[1]],
 angle[5] = π Convert[315 Degree, Radian][[1]]}
```

$\{\frac{\pi}{6}, \frac{17\,\pi}{36}, \frac{35\,\pi}{36}, \frac{3\,\pi}{2}, \frac{7\,\pi}{4}\}$

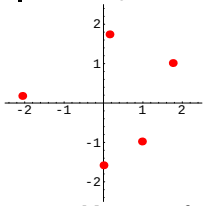
Using circular parmetrization, you get the following table of points the edge of the cam should pass through:

```
Clear[point, k]
point[k_] := displacement[k] {Cos[angle[k]], Sin[angle[k]]}

points = Table[point[k], {k, 1, 5}]
```

```
{{1.75803, 1.015}, {0.152523, 1.74334}, {-2.06212, 0.180412},
 {0, -1.58}, {0.975807, -0.975807}}
```

And a plot:

```
pointplot =
ListPlot[points, PlotStyle -> {PointSize[0.04], Red},
  AspectRatio -> Automatic,
  AxesOrigin -> {0, 0},
  PlotRange -> {{-2.5, 2.5}, {-2.5, 2.5}}];
```



Your problem is to find a parametric description of a cam whose edge passes through these points.

□Answer:

Here's an idea first suggested by John Baptiste Fourier, a French mathematician from the 1800s who pioneered the notion that functions can be made from Sine and Cosine waves the same way that music is made from basic harmonics.

You want to find a function r[t] so that

r[angle[k]] = displacement[k]

for each k = 1, 2, 3, 4, and 5.

Here's the data:

```
Clear[k]
data = Table[{angle[k], displacement[k]}, {k, 1, 5}]
```

$\{\{\frac{\pi}{6}, 2.03\}, \{\frac{17\pi}{36}, 1.75\}, \{\frac{35\pi}{36}, 2.07\}, \{\frac{3\pi}{2}, 1.58\}, \{\frac{7\pi}{4}, 1.38\}\}$

Because you have five data points, you have five degrees of freedom; so you use Fourier's idea, incorporating one constant function, two Sine waves, and two Cosine waves:

```
Clear[r, t]
r[t_] = Fit[data, {1, Cos[t], Sin[t], Cos[2 t], Sin[2 t]}, t]
```

```
1.8067 - 0.148043 Cos[t] + 0.160215 Cos[2 t] + 0.0664838 Sin[t] +
 0.275005 Sin[2 t]
```

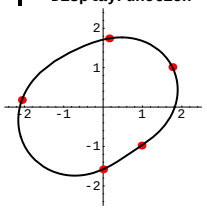
Here is the parametric formula for the curve you are after:

```
Clear[x, y]
{x[t_], y[t_]} = r[t] {Cos[t], Sin[t]}
```

```
{Cos[t] (1.8067 - 0.148043 Cos[t] +
    0.160215 Cos[2 t] + 0.0664838 Sin[t] + 0.275005 Sin[2 t]),
 Sin[t] (1.8067 - 0.148043 Cos[t] + 0.160215 Cos[2 t] + 0.0664838 Sin[t] +
    0.275005 Sin[2 t])}
```

And a plot:

```
camcurve =
 ParametricPlot[Evaluate[{x[t], y[t]}],
   {t, 0, 2 π}, PlotStyle -> Thickness[0.01],
    DisplayFunction -> Identity];

Show[pointplot, camcurve,
  AspectRatio -> Automatic,
  AxesOrigin -> {0, 0},
  PlotRange -> {{-2.5, 2.5}, {-2.5, 2.5}},
  DisplayFunction -> $DisplayFunction];
```



Damn good.

Fourier was no fool.

T.3) Parametric plotting of the predator-prey model

Remember the predator-prey model from the previous lesson. In case you've forgotten about it, here's the scoop again:

```
This mathematical model was orginally studied by Lotka and Volterra.
For a critical analysis of it, get hold of J. D. Murray, Mathematical
        Biology, Springer-Verlag, New York,1990 and read.
```

Two species coexist in a closed environment. One species, the predator, feeds on the other, the prey. There is always plenty of food for the prey, but the predators eat nothing but the prey.

Put

pred[t] = population of predators at time t.

Put

prey[t] = population of prey at time t.

It's reasonable to assume that there are positive constants a and b such that:

prey'[t] = a prey[t] − b prey[t] pred[t]

because the abundance of food for the prey allows the birth rate of the prey to be proportional to their current number, and the death rate of prey is proportional to both the current number of prey and the current number of predators.

It also makes some sense to assume that there are positive constants c and d such that:

pred'[t] = −c pred[t] + d pred[t] prey[t]

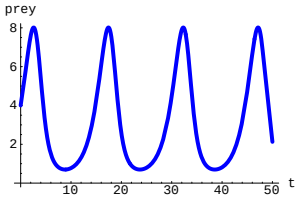
because it's reasonable to assume that the death rate of the predators is likely to be proportional to the current population of predators and that the birth rate of the predators is proportional to both the current number of the predators and the size of the food supply (the prey).

Here is what happens for a sample choice of a, b, c, d with the prey starting off with a population of 4 units and the predators starting out with a population of 1 unit.

```
Here the units could be thousands or millions so that a fractional
                    population can make sense.
```

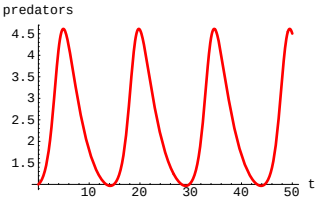
Here is *Mathematica*'s fake plot of the prey population as a function of time t for the first 50 time units:

```
endtime = 50;

a = 0.7;
b = 0.3;
c = 0.3;
d = 0.1;

Clear[pred, prey, t, Derivative]
approxsolutions =
NDSolve[{prey'[t] == a prey[t] - b prey[t] pred[t],
    pred'[t] == -c pred[t] + d pred[t] prey[t],
    prey[0] == 4,
    pred[0] == 1},
 {prey[t], pred[t]}, {t, 0, endtime}];

Clear[fakepred, fakeprey]
fakepred[t_] = pred[t] /. approxsolutions[[1]];

fakeprey[t_] = prey[t] /. approxsolutions[[1]];

preyplot =
Plot[fakeprey[t], {t, 0, endtime},
  AspectRatio -> 1/GoldenRatio,
   PlotStyle -> {{Blue, Thickness[0.015]}},
   AxesLabel -> {"t", "prey"}];
```



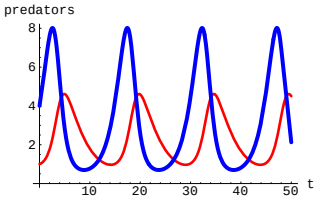
And the predator population as a function of time t:

```
predatorplot =
Plot[fakepred[t], {t, 0, endtime},
  AspectRatio -> 1/GoldenRatio,
   PlotStyle -> {{Red, Thickness[0.01]}},
   AxesLabel -> {"t", "predators"}];
```



Here they are together:

```
both = Show[predatorplot, preyplot];
```




```
The plot of the prey population is thicker than the plot of the predator
                        population.
```

Note the relationship between the predator population's crests and dips and prey population's crests and dips. Just after the predators start growing, the prey is just about eaten up.

□T.3.a)

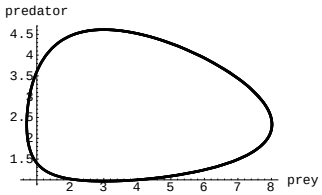
Use parametric plotting to display the predator population as a function of the prey population and describe what you see.

□Answer:

Here it is:

```
fullplot =
```

```
ParametricPlot[{fakeprey[t], fakepred[t]},
  {t, 0, endtime}, PlotStyle -> Thickness[0.01],
  PlotRange -> All,
  AspectRatio -> 1/GoldenRatio,
  AxesLabel -> {"prey", "predator"}];
```

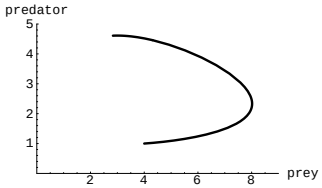


A distorted ellipse that some folks call a closed curve.

This is a little hard to interpret.

To get a grasp on it, see what happens as t goes from 0 to 5:

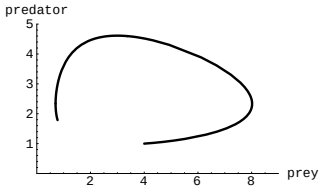
```
five = ParametricPlot[{fakeprey[t], fakepred[t]},
  {t, 0, 5}, PlotStyle -> Thickness[0.01],
  AspectRatio -> 1/GoldenRatio,
  AxesLabel -> {"prey", "predator"},
  PlotRange -> {{0, 9}, {0, 5}}];
```



Hmm.

Now see what happens as t goes from 0 to 10:

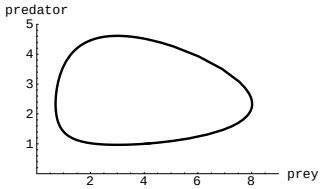
```
ten = ParametricPlot[{fakeprey[t], fakepred[t]},
  {t, 0, 10}, PlotStyle -> Thickness[0.01],
  AspectRatio -> 1/GoldenRatio,
  AxesLabel -> {"prey", "predator"},
  PlotRange -> {{0, 9}, {0, 5}}];
```



Hmmm.

Now see what happens as t goes from 0 to 15:

```
fifteen = ParametricPlot[{fakeprey[t], fakepred[t]},
  {t, 0, 15}, PlotStyle -> Thickness[0.01],
  AspectRatio -> 1/GoldenRatio,
  AxesLabel -> {"prey", "predator"},
  PlotRange -> {{0, 9}, {0, 5}}];
```

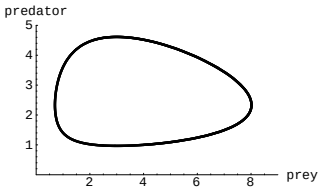


To get the full effect, grab all three plots, animate, and run forward.

Holy smoke!

This is the same as:

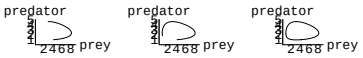
```
Show[fullplot, PlotRange → {{0, 9}, {0, 5}}];
```



And this plot shows what happens as t goes from 0 to 50.

Review the plots:

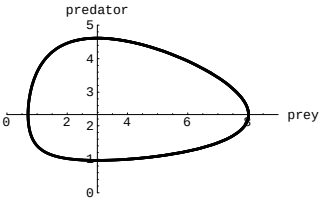
```
Show[GraphicsArray[{five, ten, fifteen}]];
```



Evidently as time advances the parametric points {prey[t], pred[t]} advance around the closed curve in a counterclockwise way. As time advances on and on, the parametric points {prey[t], pred[t]} cycle around this curve over and over again.

Take another look but this time setting the AxesOrigin → $\{\frac{c}{d}, \frac{a}{b}\}$:

```
Show[fullplot, PlotRange -> {{0, 9}, {0, 5}}, AxesOrigin -> {c/d, a/b}];
```



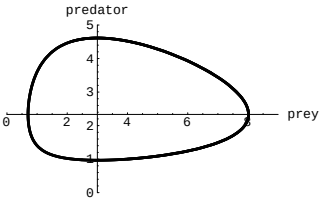
Again the closed curve tells you that predator and prey populations are periodic; they go through repeated cycles. And the time cycle for each is the same. The four sectors above defined by the axes indicate four phases depicting trends that reverse themselves as the curve crosses the axes at prey = $\frac{c}{d}$ and predator = $\frac{a}{b}$.

□T.3.b.i)

Wait a minute!

Take another look at the last plot.

```
Running this plot successfully requires that the cells in part a) are
                                active.
Show[fullplot, PlotRange → {{0, 9}, {0, 5}},
 AspectRatio -> 1/GoldenRatio, AxesOrigin -> {c/d, a/b}];
```



Those axes pierce the curve at the maximum and minimum values of prey and predator populations.

Is this just an accident?

□Answer:

Stupid question.

In mathematics, there are no accidents.

□T.3.b.ii)

Explain why those axes pierce the plot at the maximum and minimum values of prey and predator populations.

□Answer:

Look at the original differential equations

prey'[t] = a prey[t] − b prey[t] pred[t],

pred'[t] = −c pred[t] + d pred[t] prey[t].

Take the first:

prey'[t] = a prey[t] − b prey[t] pred[t].

At the times t at which the prey population is at its maximum or at its minimum, you gotta have

prey'[t] = 0.

Consequently, at the times t at which the prey population is at its maximum or at its minimum,

0 = a prey[t] − b prey[t] pred[t],

= prey[t] (a − b pred[t]).

This tells you that at the times t at which the prey population is at its maximum or at its minimum,

pred[t] = $\frac{a}{b}$.

Similarly, use

pred'[t] = −c pred[t] + d pred[t] prey[t]

to see that for a maximum or minimum predator population, you have

0 = −c pred[t] + d pred[t] prey[t]

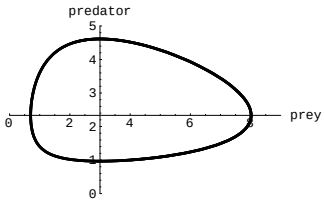
= pred[t] (−c + d prey[t]),

so when the predator population is largest or smallest,

prey[t] = $\frac{c}{d}$.

This explains why when you use AxesOrigin → {$\frac{c}{d}$, $\frac{a}{b}$}, the axes pierce the plot at the maximum and minimum values of prey and predator populations.

```
Show[fullplot,
 PlotRange -> {{0, 9}, {0, 5}}, AxesOrigin -> {c/d, a/b}];
```



Math happens again.

T.4) Quick calculations

□T.4.a)

What is the instantaneous growth rate of the parametric curve

x = x[t] = e^t + t

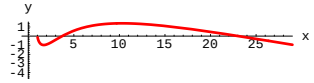
y = y[t] = t (3 − t) Log[t]

at the point at which x = 5?

□Answer:

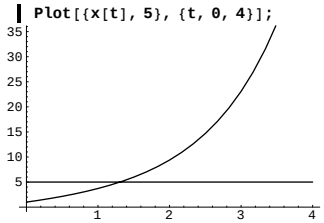
Take a look:

```
Clear[x, y, t]
x[t_] = E^t + t;
y[t_] = t (3 - t) Log[t];

curveplot =
  ParametricPlot[{x[t], y[t]}, {t, 0.01, 5},
  PlotStyle -> {Thickness[0.01], Red},
  AspectRatio -> Automatic,
  AxesLabel -> {"x", "y"}];
```



Now find the t for which x[t] = 5:

```
Plot[{x[t], 5}, {t, 0, 4}];
```



We start the search near t = 1.2:

```
FindRoot[x[t] == 5, {t, 1.2}]
{t → 1.30656}
```

The instantaneous growth rate at x = 5 is:

```
y'[t]/x'[t] /. t -> 1.30656
0.382851
```

□T.4.b)

What is the highest point on the parametric curve

x = x[t] = e^t + t

y = y[t] = t (3 − t) Log[t]?

□Answer:

Evidently the curve stays negative for larger t's because

y[t] = t (3 − t) Log[t]

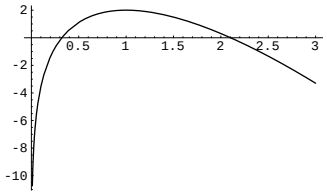
is negative for t ≥ 3.

We search for a point t with 0 < t ≤ 3 at which $\frac{dy}{dx}$ = 0.

At {x[t], y[t]}, we know $\frac{dy}{dx}$ is given by $\frac{y'[t]}{x'[t]}$.

So we have to find the t's with 0 < t ≤ 3 at which y'[t] = 0.

```
Plot[y'[t], {t, 0.01, 3}];
```



```
FindRoot[y'[t] == 0, {t, 0.3}]
{t → 0.32105}
FindRoot[y'[t] == 0, {t, 2.1}]
{t → 2.10316}
```

Look at:

```
y[0.32105]
-0.977184
y[2.10316]
1.40228
```

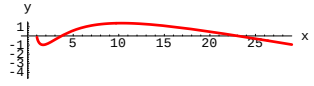
The high point:

```
high = {x[2.10316], y[2.10316]}
{10.2952, 1.40228}
```

is the highest point on the curve.

Check:

```
pointplot = Graphics[RGBColor[1, 0, 0], PointSize[0.02], Point[high]];
Show[curveplot, pointplot];
```



Nailed it.

□T.4.c)

If you have x[t] and y[t]as follows,

x'[t] = Cos[t] y[t] with x[0] = 5

y'[t] = 4 Sin[t] y[t] with y[0] = 9

then what is $\frac{dy}{dx}$ when t = π?

□Answer:

When t = π, then

$$\frac{dy}{dx} = \frac{y'[\pi]}{x'[\pi]}$$

$$\frac{=4\, y[\pi]\, \mathrm{Sin}[\pi]}{(y[\pi]\, \mathrm{Cos}[\pi])}$$

$$\frac{=4\, \mathrm{Sin}[\pi]}{\mathrm{Cos}[\pi]}$$

$$= -\frac{40}{1} = 0$$

1.09 Parametric Plotting
Give it a try!

Experience with the starred problems will be especially beneficial for understanding later lessons.

G.1) Quick calculations

□G.1.a)

Plot the curve {x[t], y[t]} with

x[t] = $\frac{3t}{1+t^3}$ and y[t] = $\frac{3t^2}{1+t^3}$

for 1 ≤ t ≤ 9.

If you regard y to be a function f[x] of x, then which x's give you f'[x] = 0?

Which x gives rise to the largest value of f[x]?

□G.1.b)

When you go with

x = x[t] = r t − r Sin[t]

y = y[t] = r − r Cos[t],

as in the Basics, you get a cycloid.

Calculating y in the form y = f[x] is almost out of the question.

Nevertheless, it's not a big deal to calculate f'[x] when you are given a specific value of x.

Here is a table in the form {x, f'[x]} for selected x's.

```
Clear[r, t, f]
Table[{N[r t - r Sin[t]], f'[N[r t - r Sin[t]]]}, {t, 0, 2 π, π/6}]
```

```
{{0, f'[0]}, {0.0235988 r, f'[0.0235988 r]}, {0.181172 r, f'[0.181172 r]},
 {0.570796 r, f'[0.570796 r]}, {1.22837 r, f'[1.22837 r]},
 {2.11799 r, f'[2.11799 r]}, {3.14159 r, f'[3.14159 r]},
 {4.16519 r, f'[4.16519 r]}, {5.05482 r, f'[5.05482 r]},
 {5.71239 r, f'[5.71239 r]}, {6.10201 r, f'[6.10201 r]},
 {6.25959 r, f'[6.25959 r]}, {6.28319 r, f'[6.28319 r]}}
```

Unfortunately f'[x] has not been calculated simply because no clean formula for f[x] is available.

Copy, paste, and edit the last instruction so that each second slot exhibits the actual value of f'[x] for the corresponding x in the first slot.

□G.1.c)

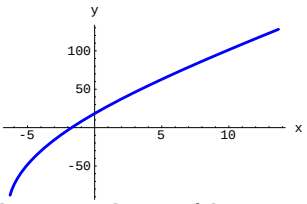
A curve is given in parametric form by

x = x[t] = $t^2 + 5t - e^{-.3t^2}$

y = y[t] = $3t^2 + 54t + 8$.

Here's a plot of part of this curve:

```
Clear[x, y, t]
x[t_] = t^2 + 5 t - E^(-0.3 t^2);
y[t_] = 3 t^2 + 54 t + 8;

ParametricPlot[{x[t], y[t]}, {t, -2, 2},
  AspectRatio -> 1/GoldenRatio,
  PlotStyle -> {{Thickness[0.01], Blue}},
  AxesLabel -> {"x", "y"}];
```



Plot more and more of the curve until you get a pretty good idea of how this baby looks.

Use information from x'[t] to give a good estimate of the smallest value of x on the curve.

Use information from y'[t] to give a good estimate of the smallest value of y on the curve.

G.2) Parametric plotting of circles and ellipses in two dimensions*

A handy way to plot the circle

$x^2 + y^2 = 1$

is to write

x[t] = Cos[t]

and

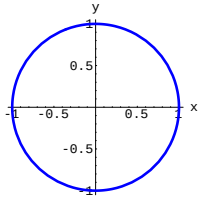
y[t] = Sin[t]

and then to plot the points

{x[t], y[t]} = {Cos[t], Sin[t]}

as t advances from 0 to 2π.

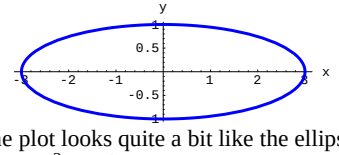
```
Clear[x, y, t]
{x[t_], y[t_]} = {Cos[t], Sin[t]};

circle =
  ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
    PlotStyle -> {{Blue, Thickness[0.015]}},
    AspectRatio -> Automatic, AxesLabel -> {"x", "y"}];
```



The option AspectRatio → Automatic tells *Mathematica* to try to give a true scale plot.

See what happens when you multiply x[t] = Cos[t] by 3:

```
Clear[x, y, t]
{x[t_], y[t_]} = {3 Cos[t], Sin[t]};

ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
   PlotStyle -> {{Blue, Thickness[0.01]}},
   AspectRatio -> Automatic, AxesLabel -> {"x", "y"}];
```



The plot looks quite a bit like the ellipse

$(\frac{x}{3})^2 + y^2 = 1$.

Check this out:

```
Clear[x, y, t]
Expand[(x/3)^2 + y^2 /. {x -> 3 Cos[t], y -> Sin[t]}, Trig -> True]
```

```
1
```

The plot **IS** the ellipse

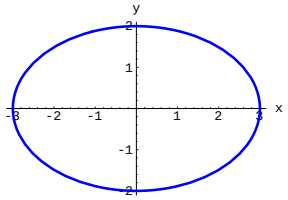
$(\frac{x}{3})^2 + y^2 = 1$.

The upshot:

{x[t], y[t]} = {3 Cos[t], Sin[t]}
gives you a parametric formula for the ellipse
$(\frac{x}{3})^2 + y^2 = 1$.

Now see what happens when you also multiply y[t] = Sin[t] by 2:

```
Clear[x, y, t]
{x[t_], y[t_]} = {3 Cos[t], 2 Sin[t]};

ParametricPlot[{x[t], y[t]} , {t, 0, 2 Pi},
   PlotStyle -> {{Blue, Thickness[0.01]}},
   AspectRatio -> Automatic, AxesLabel -> {"x", "y"}];
```



The plot looks quite a bit like the ellipse
$(\frac{x}{3})^2 + (\frac{y}{2})^2 = 1$.
Check this out:

```
Clear[x, y, t]
Expand[(x/3)^2 + (y/2)^2 /. {x -> 3 Cos[t], y -> 2 Sin[t]}, Trig -> True]
```

```
1
```

The plot **IS** the ellipse
$(\frac{x}{3})^2 + (\frac{y}{2})^2 = 1$.
The upshot:
{x[t], y[t]} = {3 Cos[t], 2 Sin[t]}
gives you a parametric formula for the ellipse
$(\frac{x}{3})^2 + (\frac{y}{2})^2 = 1$.

□G.2.a.i)

Give, as above, parametric formulas for the ellipse
$(\frac{x}{4})^2 + (\frac{y}{3})^2 = 1$
and use your formulas to plot the curve.

□G.2.a.ii)

Give, as above, parametric formulas for the ellipse
$(\frac{x}{2})^2 + (\frac{y}{4})^2 = 1$
and use your formulas to plot the curve.

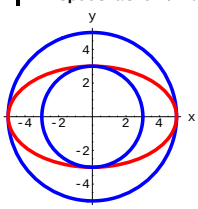
□G.2.a.iii)

Given positive constants a and b, give as above parametric formulas for the ellipse
$(\frac{x}{a})^2 + (\frac{y}{b})^2 = 1$.

□G.2.a.iv)

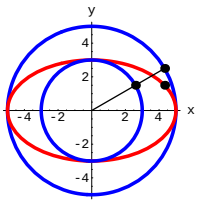
Look at this graphic:

```
a = 5;
b = 3;
Clear[acircleplotter, bcircleplotter, ellipseplotter, t]
acircleplotter[t_] = {a Cos[t], a Sin[t]};
ellipseplotter[t_] = {a Cos[t], b Sin[t]};
bcircleplotter[t_] = {b Cos[t], b Sin[t]};

CBS =
 ParametricPlot[
 {acircleplotter[t],
   ellipseplotter[t],
     bcircleplotter[t]},
 {t, 0, 2 Pi},
 PlotStyle -> {{Blue, Thickness[0.02]},
      {Red, Thickness[0.02]},
      {Blue, Thickness[0.02]}},
  AspectRatio -> Automatic, AxesLabel -> {"x", "y"}];
```



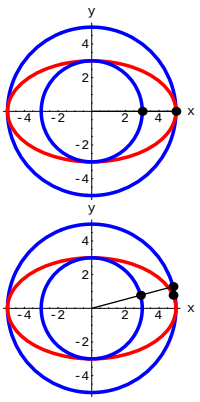
Here are the points plotted on each curve when you go with $t = \frac{\pi}{6}$:

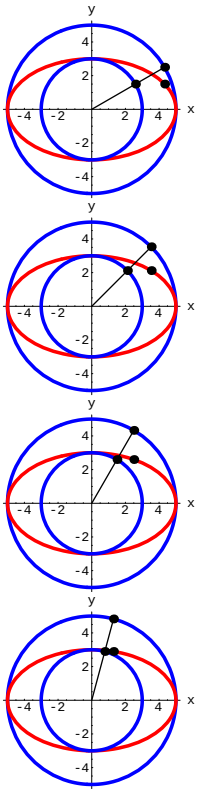
```
Clear[points, ray, t]
points[t_] = Graphics[{PointSize[0.05],
       Point[{a Cos[t], a Sin[t]}],
       Point[{a Cos[t], b Sin[t]}],
       Point[{b Cos[t], b Sin[t]}]}];
ray[t_] = Graphics[
       Line[{{0, 0}, {a Cos[t], a Sin[t]}}]];

Show[CBS, points[Pi/6], ray[Pi/6]];
```

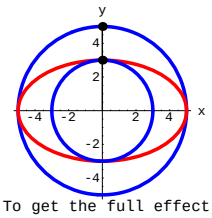


Here is how the two circles and the ellipse plot out as t runs from 0 to $\frac{\pi}{2}$:

```
Table[Show[CBS, points[t], ray[t]], {t, 0, Pi/2, Pi/12}];
```







To get the full effect, grab the plots, press command-Y, and run forward at a revealing speed.

Your friend has no machine, but has a ruler, a compass, a T-square, and a triangle. Tell your friend how to plot a lot of points on the ellipse

$$(\frac{x}{4})^2 + (\frac{y}{2})^2 = 1$$

by drawing concentric circles and then moving horizontally or vertically from well-chosen points on the circles.

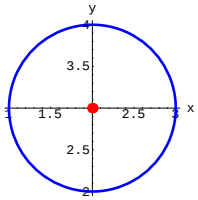
□G.2.b)

A handy way to plot the circle of radius 1 centered at {2, 3}

$$(x - 2)^2 + (y - 3)^2 = 1$$

is to use:

```
Clear[x, y, t]
{x[t_], y[t_]} = {2, 3} + {Cos[t], Sin[t]};

ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
    PlotStyle -> {{Blue, Thickness[0.015]}},
    AspectRatio -> Automatic, AxesOrigin -> {2, 3},
    AxesLabel -> {"x", "y"},
    Epilog -> {Red, PointSize[0.06], Point[{2, 3}]}];
```



Set up parametric formulas for the circle of radius r centered at {h, k} with radius r

$$(x - h)^2 + (y - k)^2 = r^2.$$

Use your formulas to plot the circle

$$(x - h)^2 + (y - k)^2 = 5.$$

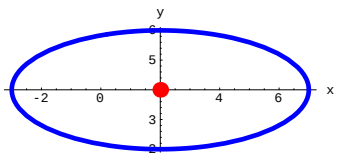
□G.2.c)

A handy way to plot the ellipse

$$(\frac{x-2}{5})^2 + (\frac{y-4}{2})^2 = 1$$

centered on {2, 4} is to use:

```
Clear[x, y, t]
{x[t_], y[t_]} = {2, 4} + {5 Cos[t], 2 Sin[t]};

ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
    PlotStyle -> {{Blue, Thickness[0.015]}},
    AspectRatio -> Automatic, AxesOrigin -> {2, 4},
    AxesLabel -> {"x", "y"},
    Epilog -> {Red, PointSize[0.05], Point[{2, 4}]}];
```



Use parametric formulas as above to plot the ellipse

$$(\frac{x+1}{2})^2 + (\frac{y-4}{5})^2 = 1.$$

At what point is this ellipse centered?

G.3) Elliptical orbits of planets and asteroids

□G.3.a) Elliptical parameters

You can plot the circle $x^2 + y^2 = r^2$ using the parametric equations

$$x[t] = r\,\text{Cos}[t]$$
$$y[t] = r\,\text{Sin}[t]$$

and running t from 0 to 2π.

You can plot the ellipse

$$(\frac{x}{a})^2 + (\frac{y}{b})^2 = r^2$$

using the parametric equations

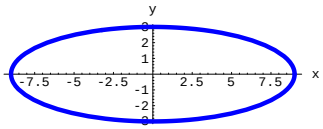
$$x[t] = a\,r\,\text{Cos}[t]$$
$$y[t] = b\,r\,\text{Sin}[t]$$

and running t from 0 to 2π.

Here you go with a true scale plot of the ellipse

$$(\frac{x}{9})^2 + (\frac{y}{3})^2 = 1:$$

```
Clear[x, y, t]
x[t_] = 9 Cos[t];
y[t_] = 3 Sin[t];

ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
  AspectRatio -> Automatic,
  PlotStyle -> {Blue, Thickness[0.015]},
  AxesLabel -> {"x", "y"}];
```



The option AspectRatio → Automatic tells *Mathematica* to try to give a true scale plot.

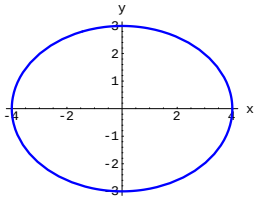
Give a true scale plot of the ellipse

$$(\frac{x}{4})^2 + (\frac{y}{9})^2 = 1.$$

□G.3.b) Eccentricity

Here is the ellipse $(\frac{x}{4})^2 + (\frac{y}{3})^2 = 1$ in true scale:

```
Clear[x, y, t]
x[t_] = 4 Cos[t];
y[t_] = 3 Sin[t];

ParametricPlot[{x[t], y[t]}, {t, 0, 2 Pi},
  AspectRatio -> Automatic,
  PlotStyle -> {Blue, Thickness[0.01]},
  AxesLabel -> {"x", "y"}];
```



The long axis of this ellipse runs along the x – axis; its length is 8.

The short axis of this ellipse runs along the y – axis; its length is 6.

Fancy folks call the long axis by the name "major axis" and they call the short axis by the name "minor axis."

To get what folks call the eccentricity of an ellipse, you take the ratio

$$\text{ratio} = \frac{\text{length of short axis}}{\text{length of long axis}}$$

and put

$$\text{eccentricity} = \sqrt{1 - \text{ratio}^2}\ .$$

Here is the eccentricity of the ellipse above:

```
long = 8;
short = 6;
ratio = short/long;

eccentricity = N[Sqrt[1 - ratio^2]]
```

```
0.661438
```

□G.3.b.i)

Plot the ellipse

$$(\frac{x}{8})^2 + (\frac{y}{3})^2 = 1$$

in true scale and calculate its eccentricity.

□G.3.b.ii)

Plot the ellipse

$$(\frac{x}{2})^2 + (\frac{y}{5})^2 = 1$$

in true scale and calculate its eccentricity.

□Tip:

Remember:

$$\text{ratio} = (\text{length of short axis}) / (\text{length of long axis})$$

and

$$\text{eccentricity} = \sqrt{1 - \text{ratio}^2}\,.$$

□G.3.b.iii)

Given positive numbers a and b, how do you know that the eccentricity of the ellipse

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

is a number no smaller than 0 but less than 1?
What do you call an ellipse whose eccentricity is 0?

□G.3.b.iv)

Plot in true scale an ellipse whose long axis is 10 units long and whose eccentricity is 0.9.

□Big tip:

Put $a = 5$ and $b = \frac{\text{short}}{2}$.

$$\text{ratio} = \frac{\text{length of short axis}}{\text{length of long axis}} \underset{\frac{=2b}{2a} = \frac{b}{a}}{=}$$

$$\text{eccentricity} = \sqrt{1 - \text{ratio}^2}$$

So

$$0.9 = \sqrt{1 - (\frac{b}{a})^2}\Big) = \sqrt{1 - (\frac{b}{5})^2}\,.$$

□G.3.b.v)

Plot in true scale an ellipse whose long axis is 10 units and whose eccentricity is 0.1.

□G.3.c) Focuses

If $a \geq b$, then the focuses of the ellipse

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

are located at the points

$$\{-\text{eccentricity}\, a,\ 0\} \text{ and } \{\text{eccentricity}\, a,\ 0\}.$$

But if $a < b$, then the focuses of the ellipse

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

are located at the points

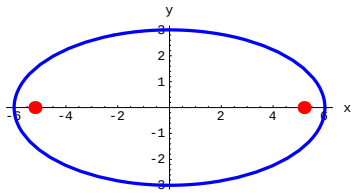
$$\{0,\ -\text{eccentricity}\, b\} \text{ and } \{0,\ \text{eccentricity}\, b\}.$$

Here is a true scale plot of the ellipse

$$(\frac{x}{6})^2 + (\frac{y}{3})^2 = 1$$

shown along with its two focuses:

```
Clear[x, y, a, b, t]
x[t_] = a Cos[t];
y[t_] = b Sin[t];
a = 6;
b = 3;

eccentricity = Sqrt[1 - (b/a)^2];

ellipse =
ParametricPlot[{x[t], y[t]},
  {t, 0, 2 Pi}, PlotStyle -> {Blue, Thickness[0.01]},
  AspectRatio -> Automatic, AxesLabel -> {"x", "y"},
  Epilog -> {{Red, PointSize[0.04],
      Point[{-eccentricity a, 0}]},
    {Red, PointSize[0.04],
      Point[{eccentricity a, 0}]}}];
```



Get it?

Plot the ellipse

$$(\frac{x}{2})^2 + (\frac{y}{9})^2 = 1$$

and show it along with its two focuses.

□Tip:

If your focuses did not plot out inside the ellipse, then you screwed up.

□G.3.d.i) Planets, asteroids and comets

Johannes Kepler (1571-1630) earned a permanent place in history by perfecting Copernicus's idea that the planets revolve about the sun. Greatly influenced by Copernicus and getting his start by working with other giants like Brahe and Galileo, he determined that planets, asteroids, and comets move in elliptical orbits around the sun with the sun at one of the focuses of the elliptical orbit.

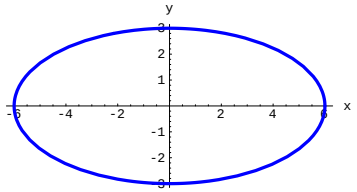
```
Physicists call this fact "Kepler's first law."
        And this is big stuff.
```

Kepler's first law tells you that to give high-quality plots of the orbits of planets, asteroids, and comets, you are well advised to plot with the sun at a focus.
Here is the usual plot of

$$(\frac{x}{7})^2 + (\frac{y}{3})^2 = 1$$

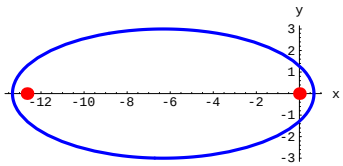
with the center of the ellipse at the origin:

```
Clear[x, y, t]
x[t_] = a Cos[t];
y[t_] = b Sin[t];
a = 7;
b = 3;

eccentricity = Sqrt[1 - (b/a)^2];

ellipse =
ParametricPlot[{x[t], y[t]},
  {t, 0, 2 Pi}, PlotStyle -> {Blue, Thickness[0.01]},
  AspectRatio -> Automatic, AxesLabel -> {"x", "y"},
  Epilog -> {{Red, PointSize[0.04],
      Point[{-eccentricity a, 0}]},
    {Red, PointSize[0.04],
      Point[{eccentricity a, 0}]}}];
```



Here is how you plot the same curve shifted to the left so that the right focus is at the origin:

```
Clear[x, y, t, a, eccentricity]
x[t_] = a Cos[t] - a eccentricity;
y[t_] = b Sin[t];

a = 7;
b = 3;

eccentricity = Sqrt[1 - (b/a)^2];

ellipse =
ParametricPlot[{x[t], y[t]},
  {t, 0, 2 Pi}, PlotStyle -> {Blue, Thickness[0.01]},
  AspectRatio -> Automatic, AxesLabel -> {"x", "y"},
  Epilog ->
    {{Red, PointSize[0.04],
      Point[{-2 eccentricity a , 0}]},
     {Red, PointSize[0.04],
      Point[{0, 0}]}}];
```



Get it?
Plot the ellipse

$$(\frac{x}{2})^2 + (\frac{y}{6})^2 = 1.$$

Then plot the same curve adjusted so that the top focus is at the origin.
Show the focuses in both plots.

□G.3.d.ii)

Plot an ellipse whose long axis is 20 units long and whose eccentricity is 0.65 with its center at the origin. Then plot the same curve adjusted so that the right focus is at the origin. Show the focuses in both plots.

□G.3.d.iii)

Kepler's first law says that planets, asteroids, and comets move in elliptical orbits around the sun with the sun at one of the focuses of the elliptical orbit.
One big mystery is that all these orbits lie roughly in the same plane. This mystery is intriguing, but the fact that all these orbits lie in one plane makes it easy to give reasonable plots of the orbits of the planets, asteroids, and comets.
Here are the data for all the planets in the form
{planet name, length of long axis of elliptical orbit, eccentricity}

```
The lengths of the long axes are given in Astronomical Units. One
 Astromonical Unit is the distance from the Earth to the Sun.
```

{Mercury, 0.774, 0.206} {Venus, 1.446, 0.007}
{Earth, 2.00, 0.017} {Mars, 3.04, 0.093}
{Jupiter, 10.4, 0.048} {Saturn, 19.08, 0.056}
{Uranus, 38.36, 0.05} {Neptune, 60.14, 0.01}
{Pluto, 78.88, 0.25}

```
Source:
William K. Hartman, Astronomy: The Cosmic Journey, Wadsworth, 1991.
```

All but Mercury and Pluto have small eccentricities; so their orbits are nearly circular. No fun plotting these.
Plot the orbit of Mercury with the sun at one focus.

□G.3.e.iv)

To get a better kick out of this, give a plot that shows the orbits of Neptune and Pluto.
Is Pluto always the outmost planet?

□Tip:

The eccentricity of Neptune's orbit is so small that you can safely plot it as a circle of radius 30.07 centered at the Sun.

□G.3.e.v)

To get an even better kick out of this, give a plot that shows the orbits of Earth and a typical member of the Apollo asteroid group.
In the form of the data above the Apollo data are {Apollo, 3.0, 0.56}.

□Tip:

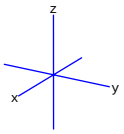
The eccentricity of Earth's orbit is so small that you can safely plot it as a circle of radius 1.00 centered at the Sun.

G.4) Parametric plotting of circles, tubes and horns in three dimensions*

Here are the three coordinate axes in three dimensions:

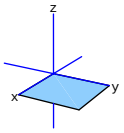
```
h = 5;

spacer = h/10;
threedims =
   Graphics3D[{
      {Blue, Line[{{-h, 0, 0}, {h, 0, 0}}]},
         Text["x", {h + spacer, 0, 0}],
      {Blue, Line[{{0, -h, 0}, {0, h, 0}}]},
         Text["y", {0, h + spacer, 0}],
      {Blue, Line[{{0, 0, -h}, {0, 0, h}}]},
         Text["z", {0, 0, h + spacer}]}];
```

```
CMView = {2.7, 1.6, 1.2};
Show[threedims, PlotRange -> All,
ViewPoint -> CMView, Boxed -> False];
```



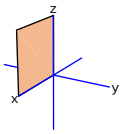
The xy-plane is the plane that you get by laying down a rigid flat sheet on the girders defined by the x-axis and the y-axis.
Here is a piece of it:

```
xyplane =
Graphics3D[
 Polygon[{{0, 0, 0}, {h, 0, 0}, {h, h, 0}, {0, h, 0}}]];

Show[threedims, xyplane, PlotRange -> All,
   ViewPoint -> CMView,
   Boxed -> False];
```



The xz-plane is the plane that you get by laying down a rigid flat sheet on the girders defined by the x-axis and the z-axis.
Here is a piece of it:

```
xzplane = Graphics3D[
 Polygon[{{0, 0, 0}, {0, 0, h}, {h, 0, h}, {h, 0, 0}}]];

Show[threedims, xzplane, PlotRange -> All,
 ViewPoint -> CMView, Boxed -> False];
```

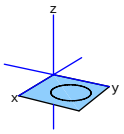


Show a piece of the yz-plane together with the coordinate axes.
Continue to use the option ViewPoint → CMView.

□G.4.a)

Here is the circle in the xy-plane of radius 1.5 centered at {2.5, 3, 0}:

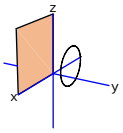
```
Clear[t]
circle = ParametricPlot3D[
   Evaluate[{2.5, 3, 0} + 1.5 {Cos[t], Sin[t], 0}],
   {t, 0, 2 Pi}, DisplayFunction -> Identity];

h = 5;
xyplane =
Graphics3D[
  Polygon[{{0, 0, 0}, {h, 0, 0}, {h, h, 0}, {0, h, 0}}]];

Show[threedims, xyplane, circle, PlotRange -> All,
   ViewPoint -> CMView, Boxed -> False,
    DisplayFunction -> $DisplayFunction];
```



Here is the circle parallel to the xz-plane of radius 1.5 centered at {2.5, 3, 2}:

```
Clear[t]
circle = ParametricPlot3D[
   Evaluate[{2.5, 3, 2} + 1.5 {Cos[t], 0, Sin[t]}],
    {t, 0, 2 Pi}, DisplayFunction -> Identity];
```

```
h = 5;
xzplane = Graphics3D[
 Polygon[{{0, 0, 0}, {0, 0, h}, {h, 0, h}, {h, 0, 0}}]];

Show[threedims, xzplane, circle, PlotRange -> All,
   ViewPoint -> CMView, Boxed -> False,
    DisplayFunction -> $DisplayFunction];
```

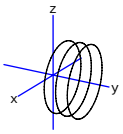


Plot the circle parallel to the yz-plane of radius 1.5 centered at {2, 3, 2}.

□G.4.b)

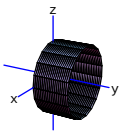
Here are several circles of radius 3 parallel to the xz-plane centered on the y-axis:

```
Clear[t]
circle1 = ParametricPlot3D[
Evaluate[{0, 1, 0} + 3 {Cos[t], 0, Sin[t]}],
     {t, 0, 2 Pi}, DisplayFunction -> Identity];

circle2 = ParametricPlot3D[
Evaluate[{0, 2, 0} + 3 {Cos[t], 0, Sin[t]}],
     {t, 0, 2 Pi}, DisplayFunction -> Identity];

circle3 = ParametricPlot3D[
Evaluate[{0, 3, 0} + 3 {Cos[t], 0, Sin[t]}],
     {t, 0, 2 Pi}, DisplayFunction -> Identity];

Show[threedims, circle1, circle2, circle3,
   ViewPoint -> CMView, Boxed -> False,
   PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```



It's no fun to settle for a few crummy circles when you can get a whole tube composed of circles of radius 3 all parallel to the xz-plane and all centered on the y-axis:

```
Clear[t, y]
tube = ParametricPlot3D[
Evaluate[{0, y, 0} + 3 {Cos[t], 0, Sin[t]}],
     {t, 0, 2 Pi}, {y, 0, 3},
    DisplayFunction -> Identity];
Show[threedims, tube, PlotRange -> All,
   ViewPoint -> CMView, Boxed -> False,
    DisplayFunction -> $DisplayFunction];
```

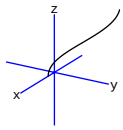


Plot a tube composed of circles of radius 3.5 all parallel to the xy-plane and all centered on the z-axis.

□G.4.c)

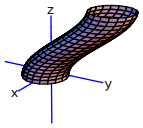
Here is a modest curve in three dimensions:

```
Clear[t, curveplotter]
curveplotter[t_] = {Cos[t], 3 t Sin[t], 3 t};

curve = ParametricPlot3D[
  Evaluate[curveplotter[t]], {t, 0, 2}, DisplayFunction -> Identity];

Show[threedims, curve, ViewPoint -> CMView,
    Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```



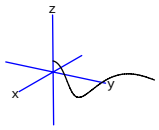
Here is what you get when you make a tube consisting of all circles parallel to the xy – plane of radius 2 centered at all points on this curve:

```
Clear[s, t]
tube = ParametricPlot3D[
  Evaluate[curveplotter[t] + 2 {Cos[s], Sin[s], 0}],
   {t, 0, 2}, {s, 0, 2 Pi},
   DisplayFunction -> Identity];

Show[threedims, tube, ViewPoint -> CMView,
    Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```



Here is another curve:

```
Clear[t, curveplotter]
curveplotter[t_] = {t Sin[t], 2 t, Cos[2 t]};

curve = ParametricPlot3D[Evaluate[curveplotter[t]],
    {t, 0, 4}, DisplayFunction -> Identity];

Show[threedims, curve, ViewPoint -> CMView,
    Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```

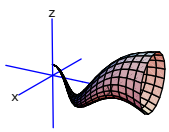


Plot the tube composed of all circles of radius 1.5 parallel to the xz – plane and centered on this curve.

□G.4.d)

Look at this plot:

```
Clear[s, t, curveplotter, rad]
curveplotter[t_] = {t Sin[t], 2 t, Cos[2 t]};

rad[t_] = t^2/6;

horn = ParametricPlot3D[
  Evaluate[curveplotter[t] +
       rad[t] {Cos[s], 0, Sin[s]}],
{t, 0, 4}, {s, 0, 2 Pi}, DisplayFunction -> Identity];

Show[threedims, horn, ViewPoint -> CMView,
  Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```



Describe the circles that this horn is composed of.

□G.4.e)

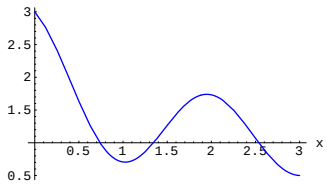
Do something that's either artistic or crazy.

G.5) Surfaces you can make by rotating curves

This is a continuation of the problem immediately above.

Here is a curve in two dimensions:

```
Clear[f, x]
f[x_] = Cos[π x] + 2 / √(x + 1);
Plot[f[x], {x, 0, 3}, PlotStyle → Blue, AxesLabel → {"x", ""}];
```



You can look at the same curve in three dimensions as a plot in the xz – plane:

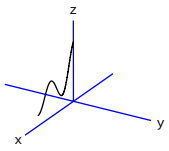
□Tip:

We'll use a shorthand way of getting the threedims axes:

```
?ThreeAxes
```

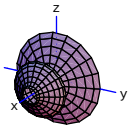
```
ThreeAxes[a,b] makes a standard cartesian axis
  graphics object with x, y, and z running
  from -a to a, and with axis labels b units
  beyond the tips of the axes.  ThreeAxes[a]
  is ThreeAxes[a,a/8].
```

```
Clear[x, curveplotter]
curveplotter[x_] = {x, 0, f[x]};

curve = ParametricPlot3D[
  curveplotter[x], {x, 0, 3}, DisplayFunction -> Identity];
threedims = ThreeAxes[4];
curveplot =

 Show[threedims, curve, PlotRange -> All,
   ViewPoint -> CMView, Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



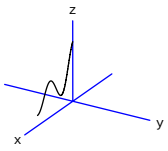
When you rotate this curve about the x – axis, you get the surface composed of all circles parallel to the yz – plane centered at {x, 0, 0} with radius f[x]:

```
Clear[s, x, surfaceplotter]
surfaceplotter[s_, x_] = {x, 0, 0} + f[x] {0, Cos[s], Sin[s]};

surface = ParametricPlot3D[
   Evaluate[surfaceplotter[s, x]],
   {x, 0, 3}, {s, 0, 2 Pi},
   DisplayFunction -> Identity];

Show[threedims, surface, ViewPoint -> CMView,
    Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```



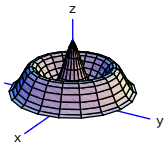
Take another look at the curve:

```
Show[curveplot];
```



When you rotate the curve about the z – axis, you get all circles parallel to the xy – plane of radius x and centered at the point {0, 0, f[x]} on the z – axis:

```
Clear[s, x, surfaceplotter]
surfaceplotter[s_, x_] =
   {0, 0, f[x]} + x {Cos[s], Sin[s], 0};
surface = ParametricPlot3D[
   Evaluate[surfaceplotter[s, x]], {x, 0, 3}, {s, 0, 2 Pi},
   DisplayFunction -> Identity];

Show[threedims, surface, ViewPoint -> CMView,
    Boxed -> False,
    PlotRange -> All,
    DisplayFunction -> $DisplayFunction];
```

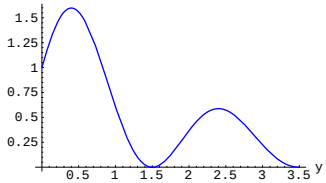


Beautiful tool, this *Mathematica* is.

□G.5.a)

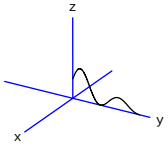
Here's another curve in two dimensions:

```
Clear[f, y]
f[y_] = E^(-0.5 y) (Sin[π y] + 1);

Plot[f[y], {y, 0, 3.5}, PlotStyle -> Blue, AxesLabel -> {"y", ""}];
```



You can look at the same curve in three dimensions as a plot in the yz – plane:

```
Clear[y, curveplotter]
curveplotter[y_] = {0, y, f[y]};

curve = ParametricPlot3D[
  curveplotter[y], {y, 0, 3.5}, DisplayFunction -> Identity];
threedims = ThreeAxes[4];

curveplot =
 Show[threedims, curve, PlotRange -> All,
   ViewPoint -> CMView, Boxed -> False,
   DisplayFunction -> $DisplayFunction];
```



Plot the surface you get when you rotate this curve about the y – axis. Follow it up with a plot of what you get when you rotate this curve about the z – axis.
Clearly identify each plot.

G.6) Projectile marksmanship

□G.6.a) Off the wall

On flat terrain, a projectile with an initial velocity of 190 ft/sec is fired at a 20 – foot high vertical wall 1000 feet from the tip of the cannon inclined at an angle a with the horizontal.

□G.6.a.i)

If $a = \frac{\pi}{6}$, does the projectile hit the wall?

□Tip:

The projectile falls short of the wall; you give the calculation.

For some help, see the Tutorials.

□G.6.a.ii)

For approximately what angles of inclination will the projectile hit the wall or soar over the wall?

□Tip:

Plot the horizontal range as a function of a.

For more on the horizontal range, see the Tutorials.

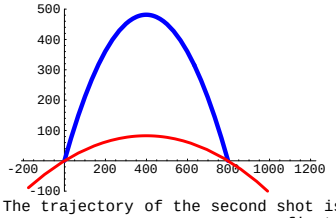
□G.6.a.iii)

For approximately what angles of inclination will the projectile hit the wall?

□G.6.b.i) Trick shots

You have a cannon with a muzzle speed of 190 ft/sec.
At time $t = 0$, you fire the cannon with an angle of inclination $\frac{\pi}{4} + z$.
Then you wait d seconds and fire again with an angle of inclination $\frac{\pi}{4} - z$.
Here is a plot which reveals what the second projectile is doing while the first is in flight in the case $z = \frac{\pi}{8}$ and $d = 1$.

```
Clear[x, y, t, a, s, a]
x[t_, s_, a_] = s t Cos[a];
y[t_, s_, a_] = -16 t^2 + s t Sin[a];

z = π/8;
d = 1;

ParametricPlot[
  {{x[t, 190, π/4 + z], y[t, 190, π/4 + z]},
  {x[t - d, 190, π/4 - z], y[t - d, 190, π/4 - z]}},
    {t, 0, 190/16 Sin[π/4 + z]},
   PlotRange -> {-100, 500},
  PlotStyle -> {{Thickness[0.015], Blue},
      {Thickness[0.010], Red}}];
```



The trajectory of the second shot is thinner than the trajectory of the first shot.

That its plot is below the ground on the right indicates that the second shot hit the ground before the first shot hit the ground.
That its plot is below the ground on the left indicates that the second shot was fired after the first shot was fired.
Your mission, if you accept it, is to find a new delay time d so that both projectiles hit the ground at the same instant.

□G.6.b.ii)

You have a cannon with a muzzle speed of s ft/sec.
At time $t = 0$, you fire a projectile with an angle of inclination $\frac{\pi}{4} + z$.
Then you wait d[z] seconds and fire another projectile with an angle of inclination $\frac{\pi}{4} - z$.
This time your mission is:
Determine a formula for the delay time $d = d[s, z]\ (\text{for } 0 < z < \frac{\pi}{4})$
as a function of z and s so that both projectiles hit the ground at the same spot and at the same instant.
Show off your formula with a couple of sharp plots.

□Tip:

You want:

```
Clear[x, y, t, a, s, z];
x[t_, s_, a_] = s t Cos[a];

y[t_, s_, a_] = -16 t^2 + s t Sin[a];

equation1 = y[t, s, π/4 + z] == 0
```

$$-16\,t^2 + s\,t\,\text{Sin}\left[\frac{\pi}{4} + z\right] == 0$$

```
Clear[d];
equation2 = y[t - d, s, π/4 - z] == 0
```

$$-16\,(-d + t)^2 + s\,(-d + t)\,\text{Sin}\left[\frac{\pi}{4} - z\right] == 0$$

Solve these two equations for t and d.

```
Solve[{equation1, equation2}, {t, d}]
```

$$\left\{\{t \to 0,\ d \to 0\},\ \left\{t \to 0,\ d \to -\frac{1}{16}\,s\,\text{Sin}\left[\frac{\pi}{4} - z\right]\right\},\right.$$
$$\left\{t \to \frac{1}{16}\,s\,\text{Sin}\left[\frac{\pi}{4} + z\right],\ d \to \frac{1}{16}\,s\,\text{Sin}\left[\frac{\pi}{4} + z\right]\right\},$$
$$\left.\left\{t \to \frac{1}{16}\,s\,\text{Sin}\left[\frac{\pi}{4} + z\right],\ d \to \frac{1}{16}\,\left(-s\,\text{Sin}\left[\frac{\pi}{4} - z\right] + s\,\text{Sin}\left[\frac{\pi}{4} + z\right]\right)\right\}\right\}$$

Now it's up to you to decide which of these to go with.

G.7) More cams

□G.7.a)

A cam is to be built with a smooth shape and such that the following central displacements occur at the given angles with the horizontal:

$30^\circ \to 3.82$
$85^\circ \to 5.00$
$175^\circ \to 1.87$
$270^\circ \to 5.00$
$315^\circ \to 4.36$

To see what this means, feed in the given information:

```
Clear[displacement]
{displacement[1] = 3.82,
 displacement[2] = 5.00,
 displacement[3] = 1.87,
 displacement[4] = 5.00,
 displacement[5] = 4.36}
```

```
{3.82, 5., 1.87, 5., 4.36}
```

```
        This instruction makes use of the Mathematica Package
Miscellaneous`Units` which is included in the intialization cells of
                              this lesson.
```

```
Clear[angle]
{angle[1] = π Convert[30 Degree, Radian][[1]],
 angle[2] = π Convert[85 Degree, Radian][[1]],
 angle[3] = π Convert[175 Degree, Radian][[1]],
 angle[4] = π Convert[270 Degree, Radian][[1]],
 angle[5] = π Convert[315 Degree, Radian][[1]]}
```

$$\left\{\frac{\pi}{6},\ \frac{17\,\pi}{36},\ \frac{35\,\pi}{36},\ \frac{3\,\pi}{2},\ \frac{7\,\pi}{4}\right\}$$

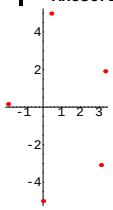
Using circular parametrization, you get the following table of points the edge of the cam should pass through:

```
Clear[point, k]
point[k_] := displacement[k] {Cos[angle[k]], Sin[angle[k]]}
points = Table[point[k], {k, 1, 5}]
```

```
{{3.30822, 1.91}, {0.435779, 4.98097}, {-1.86288, 0.162981},
 {0, -5.}, {3.08299, -3.08299}}
```

And a plot:

```
pointplot =
ListPlot[points, PlotStyle -> {PointSize[0.04], Red},
  AspectRatio -> Automatic,
  AxesOrigin -> {0, 0}];
```



Your problem is to find a parametric description of a cam whose edge passes through these points.
Show off your work by plotting your cam together with the given points.
Are you totally satisfied with the results you got?

□G.7.b)

A new cam is to be built with a smooth shape and such that the following central displacements occur at the given angles with the horizontal:

$30^\circ \rightarrow 3.82$
$85^\circ \rightarrow 5.00$
$125^\circ \rightarrow 2.05$
$175^\circ \rightarrow 1.87$
$215^\circ \rightarrow 6.02$
$265^\circ \rightarrow 1.86$
$335^\circ \rightarrow 4.36$

To see what this means, feed in the given information:

```
Clear[displacement]
{displacement[1] = 3.82,
 displacement[2] = 5.00,
 displacement[3] = 2.05,
 displacement[4] = 1.87,
 displacement[5] = 6.02,
 displacement[6] = 0.86,
 displacement[7] = 0.36}
```

```
{3.82, 5., 2.05, 1.87, 6.02, 0.86, 0.36}
```

```
       This instruction makes use of the Mathematica Package
Miscellaneous`Units` which is included in the initialization cells of
                              this lesson.
```

```
Clear[angle]
{angle[1] = π Convert[30 Degree, Radian][[1]],
 angle[2] = π Convert[85 Degree, Radian][[1]],
 angle[3] = π Convert[125 Degree, Radian][[1]],
 angle[4] = π Convert[175 Degree, Radian][[1]],
 angle[5] = π Convert[215 Degree, Radian][[1]],
 angle[6] = π Convert[265 Degree, Radian][[1]],
 angle[7] = π Convert[335 Degree, Radian][[1]]}
```

$$\left\{\frac{\pi}{6}, \frac{17\,\pi}{36}, \frac{25\,\pi}{36}, \frac{35\,\pi}{36}, \frac{43\,\pi}{36}, \frac{53\,\pi}{36}, \frac{67\,\pi}{36}\right\}$$

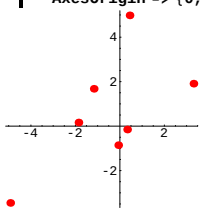
Using circular parametrization, you get the following table of points the edge of the cam should pass through:

```
Clear[point, k]
point[k_] := displacement[k] {Cos[angle[k]], Sin[angle[k]]}
points = Table[point[k], {k, 1, 7}]
```

```
{{3.30822, 1.91}, {0.435779, 4.98097}, {-1.17583, 1.67926},
 {-1.86288, 0.162981}, {-4.9313, -3.45293}, {-0.0749539, -0.856727},
 {0.326271, -0.152143}}
```

And a plot:

```
pointplot =
ListPlot[points, PlotStyle -> {PointSize[0.04], Red},
  AspectRatio -> Automatic, PlotRange -> All,
  AxesOrigin -> {0, 0}];
```



Your problem is to find a parametric description of a cam whose edge passes through these points.
Show off your work by plotting your cam together with the given points.
Are you totally satisfied with the results you got this time?
Does the positioning of the data have something to do with the outcome?

□Tip:

You have seven points so you can fit with

1, Cos[t], Sin[t], Cos[2 t], Sin[2 t], Cos[3 t], and Sin[3 t].

G.8) Parametric plotting of a predator-prey model in which the prey don't reproduce and the predators don't die

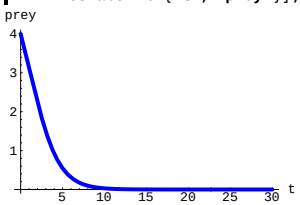
The basic predator-prey model is:

prey'[t] = a prey[t] − b prey[t] pred[t]
pred'[t] = −c pred[t] + d pred[t] prey[t].

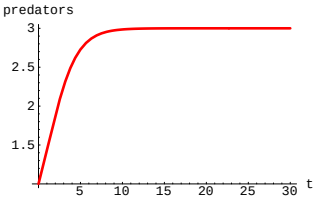
In the case in which the predators don't die and the prey don't reproduce, you have a = c = 0.
Here are some plots related to this set-up:

```
endtime = 30;

a = 0;
b = 0.2;
c = 0;
d = 0.1;

Clear[pred, prey, t, Derivative]
approxsolutions =
NDSolve[{prey'[t] == a prey[t] - b prey[t] pred[t],
    pred'[t] == -c pred[t] + d pred[t] prey[t],
    prey[0] == 4,
    pred[0] == 1},
 {prey[t], pred[t]}, {t, 0, endtime}];

Clear[fakepred, fakeprey]
fakepred[t_] = pred[t] /. approxsolutions[[1]];
```

```
fakeprey[t_] = prey[t] /. approxsolutions[[1]];

preyplot =
Plot[fakeprey[t], {t, 0, endtime},
   PlotStyle -> {{Blue, Thickness[0.015]}},
   PlotRange -> All,
   AxesLabel -> {"t", "prey"}];
```

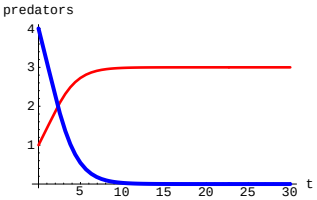



```
predatorplot =
Plot[fakepred[t], {t, 0, endtime},
   PlotStyle -> {{Red, Thickness[0.01]}},
   PlotRange -> All,
   AxesLabel -> {"t", "predators"}];
```



Here they are together:

```
both = Show[predatorplot, preyplot];
```



```
The plot of the prey population is thicker than the plot of the predator
                              population.
```

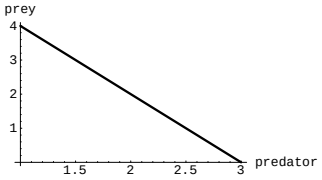
□G.8.a.i)

Interpret the plots.

□G.8.a.ii)

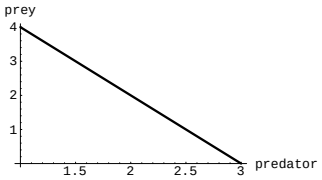
Use parametric plotting to display the predator population as a function of the prey population and describe what you see.

```
Clear[f, pred]
f[pred_] = 4 - b (pred - 1)/d
4 - 2. (-1 + pred)
Solve[f[pred] == 0]
{{pred → 3.}}
Plot[f[pred], {pred, 1, 3}, PlotStyle -> Thickness[0.01],
   PlotRange -> All,
   AxesLabel -> {"predator", "prey"}];
```



This is the same as:

```
ParametricPlot[{fakepred[t], fakeprey[t]},
 {t, 0, endtime}, PlotRange -> All,
  PlotStyle -> Thickness[0.01],
   AxesLabel -> {"predator", "prey"}];
```



You will get

$$\text{prey} = \text{prey}[0] - \frac{b}{d}(\text{pred} - \text{pred}[0])$$

for any predator-prey model with $a = c = 0$.

□G.8.a.iii) Just for your information

The model is:

$$\text{prey}'[t] = a\,\text{prey}[t] - b\,\text{prey}[t]\,\text{pred}[t]$$
$$\text{pred}'[t] = -c\,\text{pred}[t] + d\,\text{pred}[t]\,\text{prey}[t]$$

with $a = c = 0$.

You can use the chain rule to explain the linear relationship of the number of prey as a function of the number of predators.

Here's how it goes:

Put

$$\text{prey} = f[\text{pred}].$$

This gives

$$\text{prey}[t] = f[\text{pred}[t]].$$

The chain rule says

$$\text{prey}'[t] = f'[\text{pred}[t]]\,\text{pred}'[t].$$

Consequently,

$$f'[\text{pred}[t]] = \frac{\text{prey}'[t]}{\text{pred}'[t]}.$$

But

$$\text{prey}'[t] = a\,\text{prey}[t] - b\,\text{prey}[t]\,\text{pred}[t]$$
$$\text{pred}'[t] = -c\,\text{pred}[t] + d\,\text{pred}[t]\,\text{prey}[t]$$

with $a = c = 0$.

So

$$\text{prey}'[t] = -b\,\text{prey}[t]\,\text{pred}[t]$$
$$\text{pred}'[t] = d\,\text{pred}[t]\,\text{prey}[t].$$

Consequently,

$$f'[\text{pred}[t]] = \text{prey}'[t]/\text{pred}'[t] = -\frac{b}{d}.$$

This works for any t so

$$f'[\text{pred}] = -\frac{b}{d}.$$

This tells you and everyone else that

$$\text{prey} = f[\text{pred}]$$

is a line function of pred and because $\text{prey}[0] = 4$ and $\text{pred}[0] = 1$,

$$\text{prey} = f[\text{pred}]$$

is the line function with growth rate $-\frac{b}{d}$ that goes through {1, 4}.

Try it out:

G.9) Politics and the environment

□G.9.a.i)

Study the Tutorial on the predator-prey model based on the differential equations

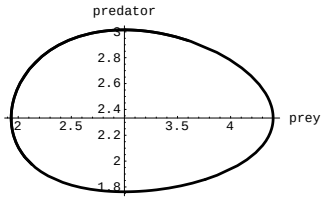
$$\text{prey}'[t] = a\,\text{prey}[t] - b\,\text{prey}[t]\,\text{pred}[t]$$
$$\text{pred}'[t] = -c\,\text{pred}[t] + d\,\text{pred}[t]\,\text{prey}[t].$$

Use the same a, b, c, and d as in the Tutorial but use starting data with

$$\text{prey}[0] = \frac{c}{d} + h,$$
$$\text{pred}[0] = \frac{a}{b} + h$$

for a relatively small value of h:

```
h = 0.6;
endtime = 20;

a = 0.7;
b = 0.3;
c = 0.3;
d = 0.1;

Clear[pred, prey, t, Derivative]
approxsolutions =
NDSolve[{prey'[t] == a prey[t] - b prey[t] pred[t],
    pred'[t] == -c pred[t] + d pred[t] prey[t],
    prey[0] == c/d + h,
    pred[0] == a/b + h},
 {prey[t], pred[t]}, {t, 0, endtime}];

Clear[fakepred, fakeprey]
fakepred[t_] = pred[t] /. approxsolutions[[1]];

fakeprey[t_] = prey[t] /. approxsolutions[[1]];

cycleplot =
 ParametricPlot[{fakeprey[t], fakepred[t]},
    {t, 0, endtime}, PlotStyle -> Thickness[0.01],
    PlotRange -> All,
    AxesLabel -> {"prey", "predator"},
    AxesOrigin -> {c/d, a/b}];
```



Why was it a good idea to set AxesOrigin → $\{\frac{c}{d}, \frac{a}{b}\}$?
Reducing the size of h has what effect on the fluctuations of the two populations? What happens when you make h = 0.01?

How about h = 0?
Try to explain, in your own terms, why this turned out the way it did.

□ **Tip:**

As you plot, pay special attention to the numbers along the axes.

□ G.9.a.ii)

Calculus&Mathematica is pleased to acknowledge that the idea for this problem came from the book Elementary Differential Equations by William Boyce and Richard DiPrima (Wiley, New York, 1977). These gentlemen wrote a really good book. You should be able to find it in your school's library.

Most folks call the populations

prey = $\frac{c}{d}$ and predators = $\frac{a}{b}$

for the predator-prey model

prey'[t] = a prey[t] − b prey[t] pred[t]
pred'[t] = −c pred[t] + d pred[t] prey[t]

that you worked with above by the name "equilibrium populations." Imagine that the prey are insects and that the predators are birds or fish. The government goes into action, spreading an insecticide like DDT all over the place. The insecticide kills both predator and prey in proportion to their current population, so the new model becomes

prey'[t] = a prey[t] − b prey[t] pred[t] − u prey[t]
pred'[t] = −c pred[t] + d pred[t] prey[t] − v pred[t]

where a, b, c, d, u, and v are all positive constants.
This is the same as

prey'[t] = (a − u) prey[t] − b prey[t] pred[t]
pred'[t] = −(c + v) pred[t] + d pred[t] prey[t]

What are the new equilibrium populations?
How do these new equilibrium populations tell you that the net effect of the government action was to raise the equilbrium population of the prey (insects) but to lower the equilibrium population of the predators? Is that what the government had in mind?

□ **Tip:**

You don't need the machine for this one.

□ G.9.b)

In the Tutorials, you saw a rationale for looking at the differential equations

prey'[t] = a prey[t] − b prey[t] pred[t]
pred'[t] = −c pred[t] + d pred[t] prey[t]

for modeling the predator-prey relationship. These differential equations are not carved in marble. In fact, current literature pays attention to trying to modify these equations to make them more realistic.

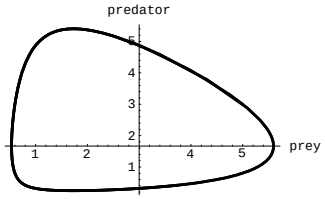
See J. D. Murray, Mathematical Biology, Springer-Verlag, New York, 1990, page 71.

For example, you can decide to replace the second equation by

pred'[t] = $-\frac{c\ \text{pred[t]}}{\text{prey[t]}}$ + d pred[t] prey[t].

Rationale:
It's reasonable to assume that the birth rate of the predators is proportional both to the current number of the predators and the size of the food supply, and the death rate of the predators is likely to be directly proportional to the current population of predators and inversely proportional to the number of prey.
Here's a try:

```
endtime = 30;
a = 0.5;
b = 0.3;
c = 0.6;
d = 0.2;

Clear[pred, prey, t, Derivative]
approxsolutions =
NDSolve[{prey'[t] == a prey[t] - b prey[t] pred[t],
    pred'[t] ==
      -c pred[t]/prey[t] + d pred[t] prey[t],
    prey[0] == 5,
    pred[0] == 3},
 {prey[t], pred[t]}, {t, 0, endtime}];

Clear[fakepred, fakeprey]
fakepred[t_] = pred[t] /. approxsolutions[[1]];

fakeprey[t_] = prey[t] /. approxsolutions[[1]];

ParametricPlot[{fakeprey[t], fakepred[t]},
    {t, 0, endtime}, PlotStyle -> Thickness[0.01],
    AspectRatio -> 1/GoldenRatio,
    PlotRange -> All,
    AxesLabel -> {"prey", "predator"},
    AxesOrigin -> {c/d, a/b}];
```



Come up with the equillibrium populations in this model and reset the axes origin accordingly.

□ Tip:

Look at

0 = a prey[t] − b prey[t] pred[t]
= prey[t] (a − b pred[t]).

And look at

0 = $-\frac{c}{\text{prey[t]}}$ + d prey[t]
$\frac{= -c + d\ \text{prey[t]}^2}{\text{prey[t]}}$.

Don't be afraid to take a square root.

G.10) Epidemics

This information was secured from the book Mathematical Biology by J. D. Murray (Springer-Verlag,1990). For a lot more on the modeling of epidemics, see this book.

The March 4, 1978 issue of the British Medical Journal reported all the details about a flu epidemic that swept through a boarding school in early 1978. Scientists were delighted that the actual data fell in line with those predicted by the SIR model.
The model for this epidemic is:

$$Sus'[t] = -a\,Sus[t]\,Inf[t],$$
$$Inf'[t] = a\,Sus[t]\,Inf[t] - b\,Inf[t]$$
$$Recov'[t] = b\,Inf[t]$$

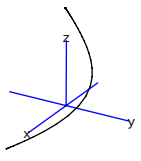
with

$$Sus[0] = 762,\ Inf[0] = 1,\ Recov[0] = 0,$$
$$a = 0.00218 \text{ and } b = 0.44036.$$

Sus[t] stands for the number of susceptible kids at time t;
Inf[t] stands for the number of infected kids at time t;
Recov[t] stands for the number of recovered kids at time t,
and t is measured in days.

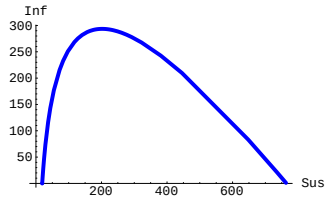
Here is a three-dimensional replay of this epidemic:

```
endtime = 30;

a = 0.00218;
b = 0.44036;

Clear[Sus, Inf, Recov, t, Derivative]
approxsolutions =
NDSolve[{Sus'[t] == - a Sus[t] Inf[t],
    Inf'[t] == a Sus[t] Inf[t] - b Inf[t],
    Recov'[t] == b Inf[t],
    Sus[0] == 763,
    Inf[0] == 1,
    Recov[0] == 0},
 {Sus[t], Inf[t], Recov[t]}, {t, 0, endtime}];

Sus[t_] = Sus[t] /. approxsolutions[[1]];

Inf[t_] = Inf[t] /. approxsolutions[[1]];

Recov[t_] = Recov[t] /. approxsolutions[[1]];

epidemicplot = ParametricPlot3D[{Sus[t], Inf[t], Recov[t]},
  {t, 0, endtime}, DisplayFunction -> Identity];
threedims = ThreeAxes[500, 20];
Show[threedims, epidemicplot, ViewPoint -> CMView,
 PlotRange -> All, Boxed -> False,
 DisplayFunction -> $DisplayFunction];
```



You can trace the progress of the epidemic starting at the lower left with lots of susceptibles and finishing at the top with lots of recovereds. In the midst of the full-blown epidemic, there are lots of infecteds.
Here's how to use a two-dimensional plot to see how the number of infecteds varies as a function of the susceptibles.

```
ParametricPlot[{Sus[t], Inf[t]}, {t, 0, endtime},
   PlotStyle -> {{Blue, Thickness[0.015]}},
   AxesLabel -> {"Sus", "Inf"}];
```



□G.10.a)

The model says

$$Inf'[t] = a\,Sus[t]\,Inf[t] - b\,Inf[t]$$
$$= (a\,Sus[t] - b)\,Inf[t].$$

Why does this tell you that the number of infecteds is the greatest when there are $\frac{b}{a}$ susceptibles?
Check this out by adding the vertical line through $\{\frac{b}{a}, 0\}$ and $\{\frac{b}{a}, 300\}$ to the plot immediately above.

□G.10.b)

Plot the susceptibles as a function of the infecteds.

□G.10.c.i)

Plot the susceptibles as a function of the recovereds.

□G.10.c.ii) Just for your information

The curve you got in part c.i) above should look like a dead ringer for exponential decay. This can be explained as follows:
Put

$$Sus = f[recov].$$

This gives

$$Sus[t] = f[recov[t]].$$

The chain rule says

$$Sus'[t] = f'[recov[t]]\,recov'[t].$$

Consequently,

$$f'[recov[t]] = \frac{Sus'[t]}{recov'[t]}.$$

But in the model,

$$Sus'[t] = -a\,Sus[t]\,Inf[t],$$
$$Recov'[t] = b\,Inf[t].$$

So

$$f'[recov[t]] = -\frac{a\,Sus[t]\,Inf[t]}{b\,Inf[t]}$$
$$= -\frac{a\,Sus[t]}{b}$$
$$= -\frac{a\,f[recov[t]]}{b},$$

because

$$f[recov[t]] = Sus[t]$$
$$f'[recov] = -\frac{a\,f[recov]}{b}.$$

This explains the exponential decay you saw above because the solution of this little differential equation is given by:

```
Clear[y, f, recov, a, b]
DSolve[{y'[recov] == -a y[recov]/b,
   y[0] == f[0]}, y[recov], recov]
```

$$\{\{y[recov] \to E^{-\frac{a\,recov}{b}}\,f[0]\}\}$$

So

$$Sus = f[recov] = f[0]\,e^{-[a/b]\,recov}.$$

Beause $Sus = 762$ when $recov = 0$,this gives

$$Sus = 762\,e^{-[a/b]\,recov}.$$

And this confirms the exponential decay.

G.11) Collision?

□G.11.a)

A Spartan missile and a Trojan missle are both flying at the same constant altitude.

At time t, the Spartan missile is at the point:

```
Clear[spartan, t]
spartan[t_] = {16.1 - 7 t + t^2, 13 t - 2 t^2}
```

```
{16.1 - 7 t + t^2, 13 t - 2 t^2}
```

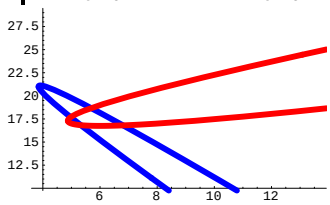
At the same time t, the Trojan missile is at the point:

```
Clear[trojan]
trojan[t_] = {26 - 13 t + 2 t^2, 23 - 5 t + t^2}
```

```
{26 - 13 t + 2 t^2, 23 - 5 t + t^2}
```

Here is a plot of the paths of the two missiles:

```
spartanpath =
  ParametricPlot[spartan[t], {t, 0, 6},
  PlotStyle -> {{Blue, Thickness[0.02]}},
  DisplayFunction -> Identity];

trojanpath =
 ParametricPlot[trojan[t], {t, 0, 6},
  PlotStyle -> {{Red, Thickness[0.02]}},
  DisplayFunction -> Identity];

Show[spartanpath, trojanpath,
  AspectRatio -> 1/GoldenRatio,
  DisplayFunction -> $DisplayFunction];
```



Their paths cross, but do they crash?

□**Tip:**

In order to crash, both missiles must be at the same point at the same time.