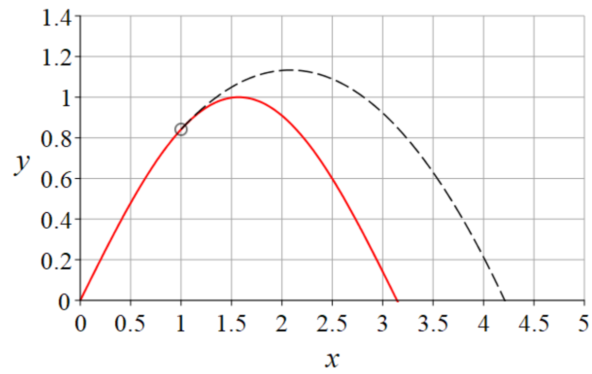
Sinetrack

Starting at $x = 0$, a ball moves along the curve:

$$y = \sin(x)$$

at constant horizontal speed u, that is its x-coordinate always increases at rate u. At some point x_0 it is released from the track and it is then acted on only by the force of gravity. The problem is to find the point of release that maximizes the distance the ball travels before landing on the x-axis.



For example, in the diagram at the right, the ball is released at $x_0 = 1$ and it lands somewhere near $x = 4.2$. The objective is to find the release point that gives us the greatest horizontal distance. That is, we want the landing point to be as far as possible from the origin.

For example, should it be released a bit later than $x = 1$? That would give us a higher starting point *and* one that already had a larger *x.* But on the down-side, the slope of the graph would be less and that means the vertical component of the velocity would be less.

On the other hand, if we released it earlier than $x = 1$, we would lose altitude but we would have a slightly higher slope and therefore a greater vertical velocity.

What might be the optimal release point x_0?

To solve the problem we will need to know the horizontal velocity u, and the acceleration due to gravity g. You probably already know that $g = 9.8$ m/s^2, but of course that assumes that the unit of distance is meters and the unit of time is seconds. If the units change, the value of g will change. In our numerical calculations here we will take:

$$g = 0.5 \quad \text{and} \quad u = 1.$$

The first thing most students want to do shoot a few balls off and see what happens. But to do that, we need an equation for the path that the ball follows after it is released. So we begin with the construction of that.

We will assume (as is standard) that the only force acting on the ball after it is released is gravity, and the effect of that is to reduce the vertical velocity of the ball at rate g. That is the upward component of the velocity of the ball is reduced by g in every unit of time.

Let's measure time t from the moment the ball leaves the origin. For now, we will keep u and g as general symbols as it helps us to understand the equations.

At the takeoff point: $t = t_0$

time:	t_0
x-velocity	u (constant)
x-coord	$x_0 = ut_0$
y-coord	$y_0 = \sin(x_0)$
Slope of graph	$m_0 = \cos(x_0)$
y-velocity	$v_0 = m_0 u = u\cos(x_0)$

Now let's describe the ball after it is released. Take a time $t > t_0$. The hard part is to find the y-coordinate. We will do this by first finding the vertical velocity v at t using the fact that it decreases at constant rate g, and then use the fact that the average velocity over the interval $[t_0, t]$ is the average of the two velocities v_0 and v.

At any time $t > t_0$

time:	t
x-velocity	u
x-coord	$x = ut$
y-velocity	$v = v_0 - g(t - t_0)$

average of v over the interval $[t_0, t]$:

$$\bar{v} = \frac{v_0 + v}{2} = v_0 - \frac{1}{2}g(t - t_0)$$

y-coord

$$y = y_0 + \bar{v}(t - t_0) = y_0 + v_0(t - t_0) - \frac{1}{2}g(t - t_0)^2$$

This last equation simply says that the vertical distance traveled $(y - y_0)$ is the average velocity times the elapsed time. That's of course *always* true no matter how the velocity changes. Thus the first part of this equation:

$$y = y_0 + \bar{v}(t - t_0)$$

is true no matter how the velocity changes. But the second part comes from the simple formula $\bar{v} = \frac{v_0+v}{2}$ and that's true only if the velocity changes at a constant rate—as it does here.

Okay—we have found the y-equation. Setting u = 1 and g = 0.5 we get a cleaner version:

$$y = y_0 + v_0(t - t_0) - \frac{1}{4}(t - t_0)^2 \qquad \begin{cases} \mathrm{y}_0 = \cos(x_0) \\ v_0 = \sin(x_0) \end{cases}$$

What's important here are not the equations *per se* but the "story" behind their construction. For example, by this stage the student should be completely comfortable with an equation of the form:

$$v = v_0 - g(t - t_0)$$

It says that we start at t_0 with velocity v_0 and then over the interval of length $t_1 - t_0$, v decreases at rate g.
This is simply a version of the general straight-line equation:

$$y = y_0 + m(x - x_0)$$

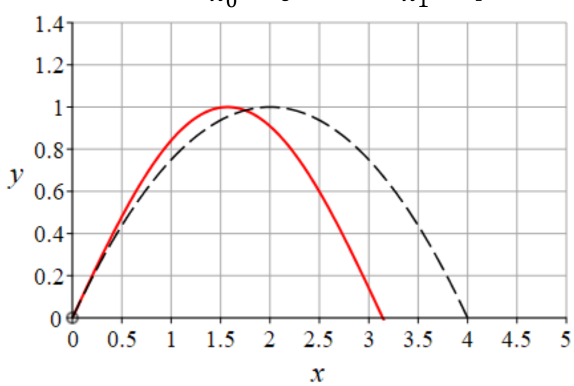
We have a formula for the height *y* of the ball when it is in free flight, after leaving the track. It gives us *y* in terms of the time *t* since it left the origin. But to plot this path in the *x-y* plane, we need *y* in terms of *x*. Since *x* = *ut* and *u* = 1, *x* and *t* are really the same. Thus our formula can be written:

$$y = y_0 + v_0(x - x_0) - \frac{1}{4}(x - x_0)^2 \qquad \begin{cases} y_0 = \cos(x_0) \\ v_0 = \sin(x_0) \end{cases}$$

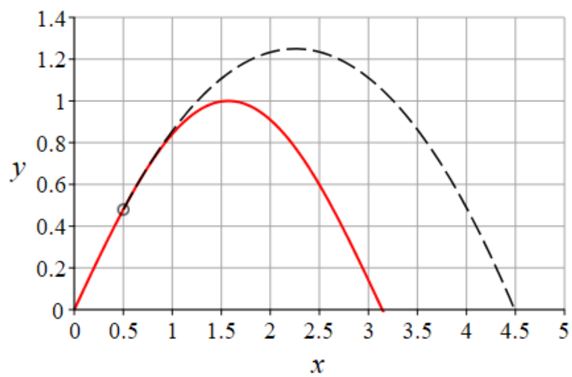
and this gives us the trajectory of the ball after it take off at x_0. Of course it depend on x_0.

I now ask the students to use technology to plot the trajectory for various values of x_0—say $x_0 = 0, 0.5, 1, 1.5$. And to use these graphs to get good estimates of the horizontal distance x_1 that the ball travels before it hits the ground. This will give them a sense of how the landing point x_1 depends on the takeoff point x_0.

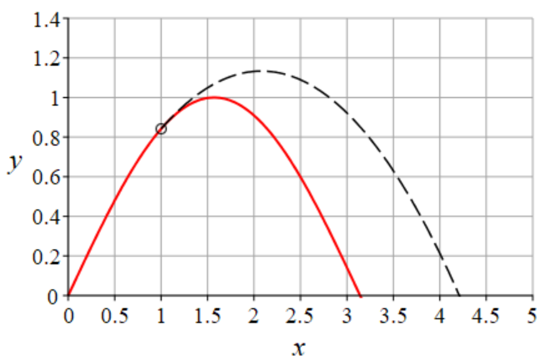
$x_0 = 0 \qquad x_1 \approx 4$



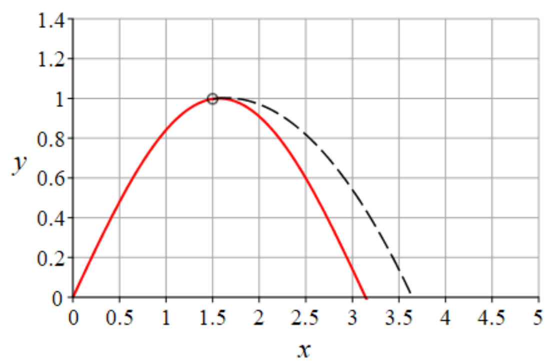
$x_0 = 0.5 \qquad x_1 \approx 4.5$



$x_0 = 1.0 \qquad x_1 \approx 4.2$



$x_0 = 1.5 \qquad x_1 \approx 3.2$



Interesting. The winner of this group of four is $x_0 = 0.5$ with a landing at $x_1 = 4.5$.

Having made these plots, I ask the students to plot what they think the graph of x_1 against x_0 might look like. See page 5 for the real graph.

And then having done that––can they use the equation above to actually get a formula for x_1 in terms of x_0. That's a good problem.

Getting a formula for x_1 in terms of x_0.
Okay––x_1 is the value of x where the ball lands so that's the x-value where $y = 0$. So if we set $y = 0$ in the equation of the trajectory:

$$y = y_0 + v_0(x - x_0) - \frac{1}{4}(x - x_0)^2$$

and also set $x = x_1$, we should get and equation involving both x_0 and x_1.

$$0 = y_0 + v_0(x_1 - x_0) - \frac{1}{4}(x_1 - x_0)^2$$

What do we have? We have a quadratic equation in $x_1 - x_0$, the horizontal distance between the takeoff point and the landing point. We can use the quadratic formula. Put everything on the left side so that the square term has positive coefficient and put the terms in the standard order:

$$\frac{1}{4}(x_1 - x_0)^2 - v_0(x_1 - x_0) - y_0 = 0$$

I choose to multiply by 4 to clear fractions:

$$(x_1 - x_0)^2 - 4v_0(x_1 - x_0) - 4y_0 = 0$$

Use the quadratic formula:

$$(x_1 - x_0) = \frac{4v_0 \pm \sqrt{16{v_0}^2 + 16y_0}}{2}$$

$$= \frac{4v_0 \pm 4\sqrt{{v_0}^2 + y_0}}{2}$$

$$= 2v_0 + \sqrt{{v_0}^2 + y_0}$$

Solve for x_1:

$$x_1 = x_0 + 2v_0 + 2\sqrt{{v_0}^2 + y_0} \qquad \begin{cases} y_0 = \sin(x_0) \\ v_0 = \cos(x_0) \end{cases}$$

This is what we wanted––a formula for the landing point x_1 in terms of the takeoff point x_0.

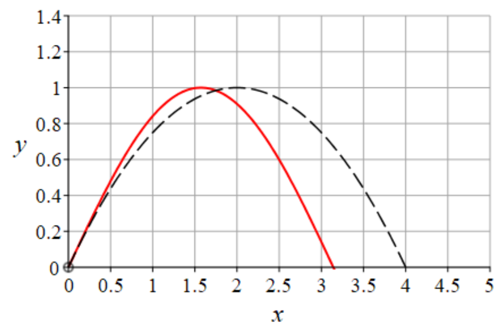
I ask the students to use technology to plot x_1 against x_0.and to clearly depict the relationship between this graph and the four trajectory graphs they have previously plotted.

This problem is an excellent example of what I mean when I say that we need to give our students more examples of working with complex systems.

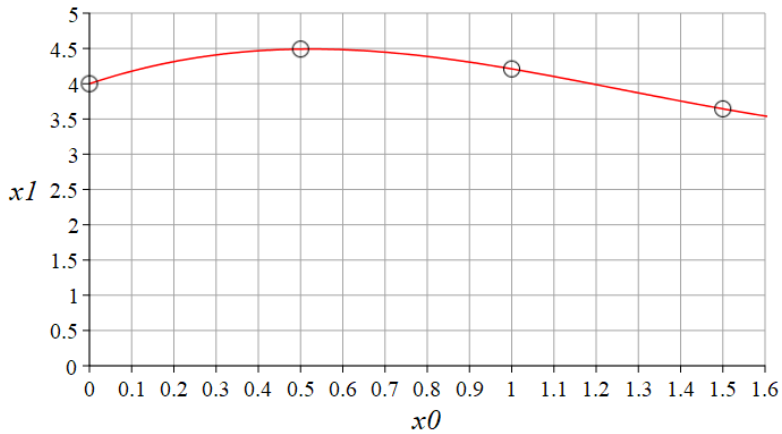
The challenge here is to keep the goal in mind and not to lose sight of the essential functional dependencies captured by the equations. This is exactly the type of thing they have difficulty with in first-year at university.

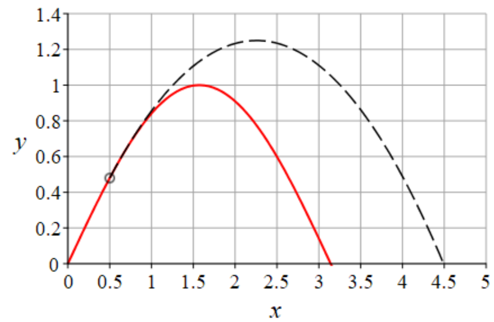
We have derived the formula giving the dependence of the landing point on the takeoff point:

$$x_1 = x_0 + 2v_0 + 2\sqrt{{v_0}^2 + y_0} \qquad \begin{cases} y_0 = \sin(x_0) \\ v_0 = \cos(x_0) \end{cases}$$

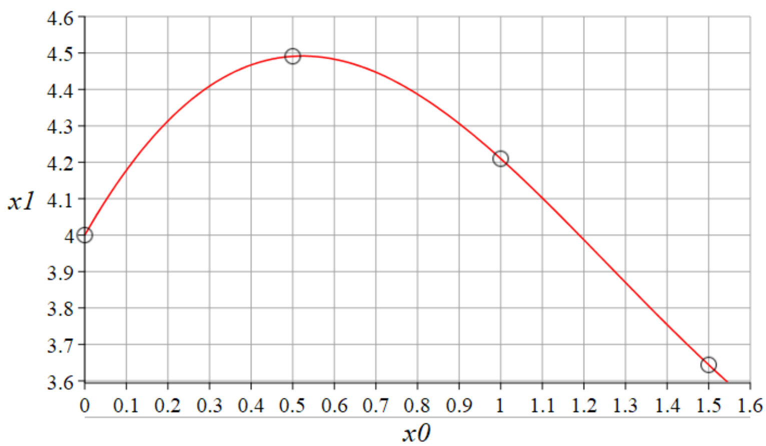


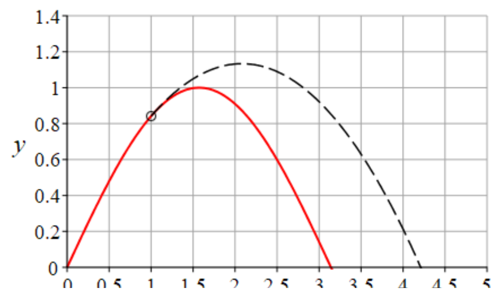
And here is the graph of the relationship.

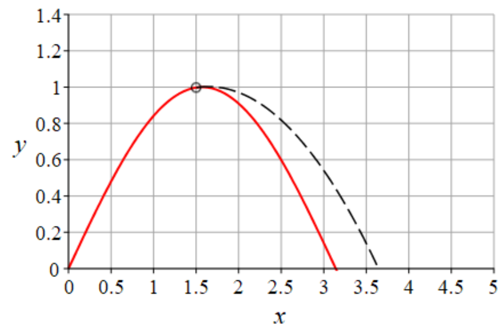




In fact we should replot to get more resolution.







We see that the optimal takeoff point is very close to $x_0 = 0.5$. It might be just a tiny bit higher.

Using calculus to find the optimal takeoff point.
The maximum value attained by x_1 will occur where its x_0-derivative is zero.

$$\frac{d}{dx_0}\left[x_0 + 2\cos(x_0) + 2\sqrt{\cos(x_0)^2 + \sin(x_0)}\right] = 0$$

We can certainly calculate the derivative but the resulting equation is hard to solve for x_0 and really needs technology. That's not a problem for us as our real objective here is on seeing the big picture and understanding the chain of ideas. Using Maple, I get the optimum at ${x_0}^* \approx 0.52$.

The final task for the students is to construct an animation of the ball for any given takeoff point x_0. Here's what the Maple program might look like.

```
> x0 := 0.5 :
> track := proc(x)
    #for a given x0, plots the trajectory of the ball starting at the origin and taking off at x0
  local baseball, ball, traj :
  global takeoff, sineplot, ypath :
  takeoff := plot( [ [x0, sin(x0) ] ], style = point, symbol = circle, symbolsize = 20, color = black ) :
  ypath := t → sin(x0) + cos(x0) · (t − x0) − 0.25 · (t − x0)^2 :
  sineplot := plot( sin(t), t = 0 ..5, color = red, thickness = 2) :
  baseball := plot( [ [x, 0] ], style = point, symbol = solidcircle, symbolsize = 10, color = black ) :
  if x < x0 then
  #for these values of x the ball is confined to the sime track
  ball := plot( [ [x, sin(x) ] ], style = point, symbol = solidcircle, symbolsize = 10, color = black ) :

  display( sineplot, takeoff, baseball, ball, view = [0 ..5, 0 ..1.4], labels = ["x", "y"], axis[1] = [gridlines
      = [0, 0.5, 1.0, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]], axis[2] = [gridlines = [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2,
      1.4, 1.6]] , axesfont = [TIMES, ROMAN, 20], labelfont = [TIMES, italic, 24]);

  else
  #for these values of x the ball follows a parabola under the influence of gravity.
  ball := plot( [ [x, ypath(x) ] ], style = point, symbol = solidcircle, symbolsize = 10, color = black ) :
  traj := plot( [t, ypath(t) , t = x0 ..x], color = black, linestyle = 3 ) :

  display( sineplot, takeoff, traj, baseball, ball, view = [0 ..5, 0 ..1.4], labels = ["x", "y"], axis[1]
      = [gridlines = [0, 0.5, 1.0, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]], axis[2] = [gridlines = [0, 0.2, 0.4, 0.6,
      0.8, 1.0, 1.2, 1.4, 1.6]] , axesfont = [TIMES, ROMAN, 20], labelfont = [TIMES, italic, 24]);
  end if
  end proc:

> plots[animate]( track, [x], x = 0 ..5, frames = 101);
```