Fri. 9/14	**2.3** Trajectory and Range with Linear Resistance	
Mon. 9/17 Tues.9/18	**2.4** Quadratic Air Resistance	**HW2b** (2.B-.F) & Friday's Handout

Email students the code

Handout / Bring:

- Computer Exercise
  - Code on computers, student laptops
- Handout on 2nd approximation to the range with linear drag (should have distributed last time)

Trajectory:

We want to describe the path or trajectory of a projectile. We could just find pairs $(x(t), y(t))$ for many times, but in this case we can find a function for $y(x)$.

In Vacuum Trajectory

For comparison, the equations in vacuum (with the *y* axis pointing upward and the same initial conditions) are:

$$x(t) = v_{xo}t,$$

$$y(t) = v_{yo}t - \tfrac{1}{2}gt^2.$$

Solving the first for the time, $t = x/v_{xo}$, and substituting that into the second gives a parabola:

$$y = v_{yo}(x/v_{xo}) - \tfrac{1}{2}g(x/v_{xo})^2,$$

$$y = \left(\frac{v_{yo}}{v_{xo}}\right)x - \left(\frac{g}{2v_{xo}^2}\right)x^2.$$

Note: by "completing the square" you can massage this into the form

$$(y - y_{peak}) = A(x - x_{peak})^2$$

Where

$$A = -\frac{g}{2v_{xo}}$$

And (not surprisingly)

$$(x_{peak}, y_{peak}) = \left(\frac{v_{y_o}, v_{x_o}}{g}, \frac{v_{y_o}^2}{2g}\right)$$ (notice that x_{peak} is ½ the range)

With Linear Drag Trajectory

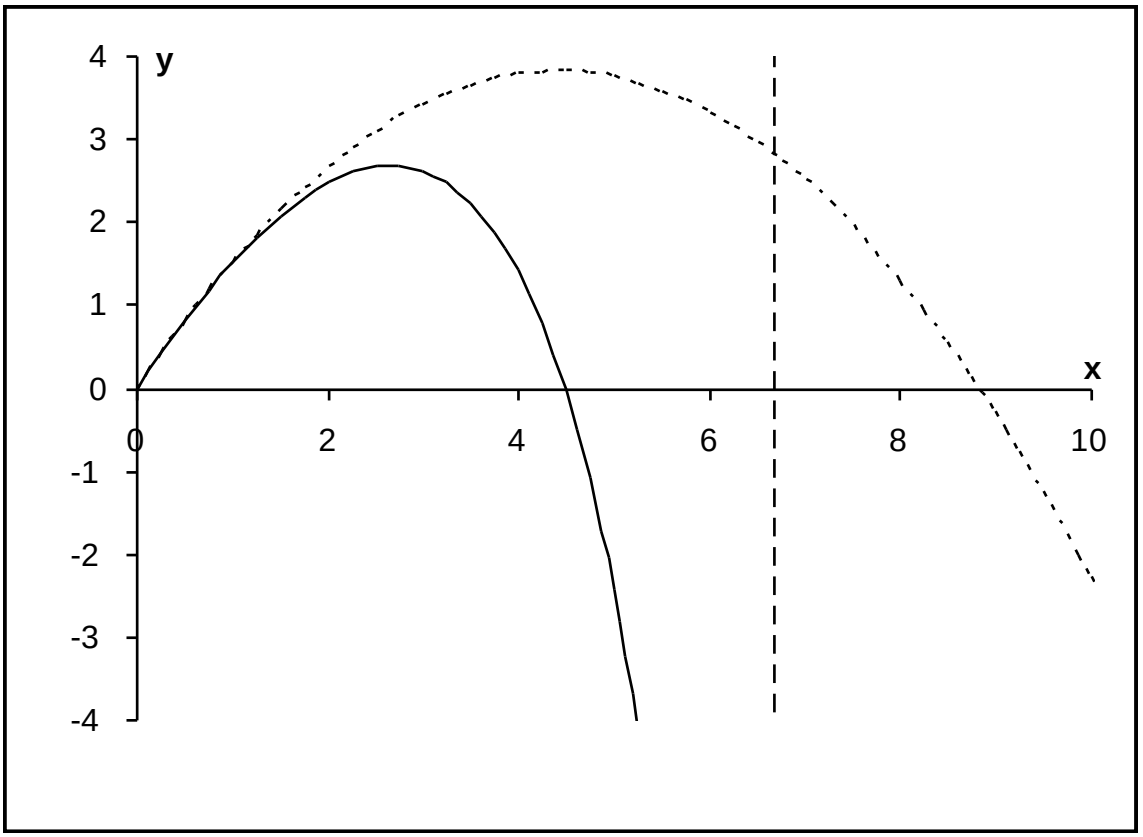
Recall that Newton's 2nd Law then takes the form

$$\vec{F}_{net} = m\vec{g} - b\vec{v}$$

Now, without doing any more analytical work than that, we can *computationally* model the flight of the projectile, it's trajectory, by the Euler-Cromer method.

Break for Coding – do part (a) of the Computational Worksheet.

Proes & cons of computational modeling. The great *advantage* of modeling computationally is that it's easy to do. A *disadvantage* is that it must be done for specific values (you can make this a tad better by defining 'unitless' parameters) so you need to run it under several conditions to be confident of any general trends.



Analytical modeling.

Last time, we found the following equations for motion in a linear medium <u>with the *y* axis pointing downward</u>:

$$x(t) = v_{xo}\tau_l\left(1 - e^{-t/\tau_l}\right),$$

$$y(t) = v_{\text{ter},l}t + \left(v_{yo} - v_{\text{ter},l}\right)\tau_l\left(1 - e^{-t/\tau_l}\right).$$

If we switch the *y* axis around so that it points upward, the only thing that changes is the sign in front of $v_{\text{ter},l}$. That is because switching around the *y* axis changes the sign in front of *g* in the differential equation and

$$v_{\text{ter},l} = \frac{mg}{b}$$
$$\tau_l = \frac{m}{b}$$

The initial component of the velocity v_{yo} must be given the appropriate sign when put in. Therefore, the equations <u>with the *y* axis pointing upward</u> are:

$$x(t) = v_{xo}\tau_l\left(1 - e^{-t/\tau_l}\right),$$
$$y(t) = \left(v_{yo} + v_{\text{ter},l}\right)\tau_l\left(1 - e^{-t/\tau_l}\right) - v_{\text{ter},l}t.$$

Solve the first of these equations for *t:*

$$1 - e^{-t/\tau_l} = x/v_{xo}\tau_l$$
$$e^{-t/\tau_l} = 1 - x/v_{xo}\tau_l$$
$$t = -\tau_l \ln\left(1 - x/v_{xo}\tau_l\right)$$

Substitute the first and third equation above into the equation for *y* to get:

$$y = \left(v_{yo} + v_{\text{ter},l}\right)\tau_l\left(x/v_{xo}\tau_l\right) + v_{\text{ter},l}\tau_l \ln\left(1 - x/v_{xo}\tau_l\right)$$

$$\boxed{y = \left(\frac{v_{yo} + v_{\text{ter},l}}{v_{xo}}\right)x + v_{\text{ter},l}\tau_l \ln\left(1 - \frac{x}{v_{xo}\tau_l}\right)}.$$

Break for coding

Implement part (b) of the worksheet. Since this is an exact relation, it should be 'right' in that, if the trajectories you drew by the Euler-Crommer method disagree – it's the latter's fault (likely too large time steps.)

Note: while this expression is transcendental, you can get expressions for the peak of the flight: take the derivative with respect to x, set equal to 0, and solve for the x value which satisfies, then plug that back into this expression for the corresponding y value.

For x, you get $\boxed{x_p = \frac{v_{xo}v_{yo}}{g}\left(\frac{1}{1 + \frac{v_{yo}}{v_t}}\right)}$;

which clearly reduces to the vacuum solution as b disappears and v_t blows up. I leave it as an exercise to the interested to find the expression for y_p.

Horizontal Range:

The range is how far away a projectile is launched from the ground will it land, assuming the ground is level. If the initial position is different or the ground is not level, this does not apply!

The range in vacuum R_{vac} is the value of x when y is zero:

$$0 = \left(\frac{v_{yo}}{v_{xo}}\right)R_{\text{vac}} - \left(\frac{g}{2v_{xo}^2}\right)R_{\text{vac}}^2 = R_{\text{vac}}\left[\left(\frac{v_{yo}}{v_{xo}}\right) - \left(\frac{g}{2v_{xo}^2}\right)R_{\text{vac}}\right],$$

so:

$$R_{\text{vac}} = \frac{2v_{xo}v_{yo}}{g}.$$

The range in a linear medium is found in the same way, but the equation is more complicated:

$$0 = \left(\frac{v_{yo} + v_{\text{ter},l}}{v_{xo}}\right)R + v_{\text{ter},l}\tau_l \ln\left(1 - \frac{R}{v_{xo}\tau_l}\right).$$

In fact, this is a transcendental equation, which means that it has no simple & closed analytical solution. What we'll aim for then is a series expansion; however, even that is hard to get.

Our approximation.

Ultimately, what we'll be working on is an expression of the form

$$R = R_0 + C_1\left(\frac{v_{oy}}{v_{ter,l}}\right) + C_2\left(\frac{v_{oy}}{v_{ter,l}}\right)^2 + C_3\left(\frac{v_{oy}}{v_{ter,l}}\right)^3 + \ldots$$

Essentially, a Taylor expansion in terms of the ratio of y-launch speed vs. terminal speed.

Our Process for Approximating

Here's how we'll go about it.

1. Expand our expression in terms of $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau}\right)$
2. Solve for R in terms of $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau}\right)$.
3. Find the approximation for when $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau}\right)$ is quite small.

4. Plug *that* approximation for R *back* into our expansion keeping the next lowest order term to find new approximate solution.
5. Return to the expression and plug *this* solution in for next higher-order terms
6. Find new approximate solution…

We can find an approximate solution to the problem if the air resistance is not too large. In that case, b is small so both:

$$v_{\text{ter},l} = \frac{mg}{b},$$

and:

$$\tau_l = \frac{m}{b},$$

are large.

Step 1: Expand our expression in terms of $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau} \right)$

If we define $\varepsilon = R/v_{xo}\tau_l$, which must be small, then we can Taylor expand the second term in the equation that we're trying to solve. We will use:

$$\ln(1-\varepsilon) = -\left(\varepsilon + \tfrac{1}{2}\varepsilon^2 + \tfrac{1}{3}\varepsilon^3 + \cdots\right),$$

and keep the first three terms to get:

$$\left(\frac{v_{yo} + v_{\text{ter},l}}{v_{xo}} \right) R - v_{\text{ter},l}\tau_l \left[\frac{R}{v_{xo}\tau_l} + \frac{1}{2}\left(\frac{R}{v_{xo}\tau_l} \right)^2 + \frac{1}{3}\left(\frac{R}{v_{xo}\tau_l} \right)^3 + \frac{1}{4}\left(\frac{R}{v_{xo}\tau_l} \right)^4 + \ldots \right] = 0.$$

We can factor out an R, which means that $R = 0$ is a solution (but not the one we want). Dividing out the factor of R and simplifying the constant terms gives:

$$\left(\frac{v_{yo}}{v_{xo}} \right) - \left(\frac{v_{\text{ter},l}}{2\tau_l v_{xo}^2} \right) R - \left(\frac{v_{\text{ter},l}}{3\tau_l^2 v_{xo}^3} \right) R^2 - \left(\frac{v_{\text{ter},l}}{4\tau_l^3 v_{xo}^4} \right) R^3 + \ldots = 0$$

Notice that $v_{\text{ter},l}/\tau_l = g$, so we can clean up a little:

$$\left(\frac{v_{yo}}{v_{xo}} \right) - \left(\frac{g}{2v_{xo}^2} \right) R - \left(\frac{g}{3\tau_l v_{xo}^3} \right) R^2 - \left(\frac{g}{4\tau_l^2 v_{xo}^4} \right) R^3 + \ldots = 0$$

.

Step 2: Solve for R in terms of $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau}\right)$.

$$R = \left(\frac{2v_{xo}^2}{g}\right)\left[\left(\frac{v_{yo}}{v_{xo}}\right) - \left(\frac{g}{3\tau_l v_{xo}^3}\right)R^2 - \left(\frac{g}{4\tau_l^2 v_{xo}^4}\right)R^3 + \ldots\right]$$

$$R = \left(\frac{2v_{xo}v_{yo}}{g}\right) - \left(\frac{2}{3\tau_l v_{xo}}\right)R^2 - \left(\frac{1}{2\tau_l^2 v_{xo}^2}\right)R^3 + \ldots$$

$$R = \left(\frac{2v_{xo}v_{yo}}{g}\right) - R\left[\frac{2}{3}\left(\frac{R}{\tau_l v_{xo}}\right) + \frac{1}{2}\left(\frac{R}{\tau_l v_{xo}}\right)^2 + \ldots\right]$$

Step 3: Find the approximation for when $\varepsilon \equiv \left(\frac{R}{v_{x0}\tau}\right)$ is quite small.

$$R \approx \boxed{R_o \equiv \left(\frac{2v_{xo}v_{yo}}{g}\right)}$$

the solution we get when there is no drag.

Step 4: Plug *that* approximation for R *back* into our expansion keeping the next lowest-order term to find next order approximation.

$$R = \left(\frac{2v_{xo}v_{yo}}{g}\right) - \left(\frac{2v_{xo}v_{yo}}{g}\right)\left[\frac{2}{3}\left(\frac{1}{\tau_l v_{xo}}\frac{2v_{xo}v_{yo}}{g}\right) + \frac{1}{2}\left(\frac{R}{\tau_l v_{xo}}\right)^2 + \ldots\right]$$

Too small to keep

Again using $v_{\text{ter},l} = \tau_l g$:

$$R \approx R_1 = R_{\text{vac}} - \left(\frac{8v_{xo}v_{yo}^2}{3\tau_l g^2}\right)^2$$

$$\boxed{R_1 = R_{\text{vac}}\left(1 - \frac{4v_{yo}}{3v_{\text{ter},l}}\right)}$$

It makes sense that this is less than the range in vacuum. The method of finding an approximate solution, then using that to find a better one is called an iterative or perturbative solution.

Step 5: Plug *that* approximation for R *back* into our expression keeping the *next* lowest-order term to find our next order approximation (see the handout)

$$\left(\frac{v_{yo}}{v_{xo}}\right)-\left(\frac{v_{ter,l}}{2\tau_l v_{xo}^2}\right)R-\left(\frac{v_{ter,l}}{3\tau_l^2 v_{xo}^3}\right)R^2-\left(\frac{v_{ter,l}}{4\tau_l^3 v_{xo}^4}\right)R^3=0,$$

or using $v_{ter,l}=\tau_l g$:

$$\left(\frac{v_{yo}}{v_{xo}}\right)-\left(\frac{g}{2v_{xo}^2}\right)R-\left(\frac{g}{3\tau_l v_{xo}^3}\right)R^2-\left(\frac{g}{4\tau_l^2 v_{xo}^4}\right)R^3=0.$$

Isolate the single power of R:

$$R=\left(\frac{2v_{xo}^2}{g}\right)\left[\left(\frac{v_{yo}}{v_{xo}}\right)-\left(\frac{g}{3\tau_l v_{xo}^3}\right)R^2-\left(\frac{g}{4\tau_l^2 v_{xo}^4}\right)R^3\right],$$

$$R=\left(\frac{2v_{xo}v_{yo}}{g}\right)-\left(\frac{2}{3\tau_l v_{xo}}\right)R^2-\left(\frac{1}{2\tau_l^2 v_{xo}^2}\right)R^3.$$

Get the improved solution by making the substitution $R\approx R_1$ on the right hand side:

$$R\approx R_2=\left(\frac{2v_{xo}v_{yo}}{g}\right)-\left(\frac{2}{3\tau_l v_{xo}}\right)R_{vac}^2\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^2-\left(\frac{1}{2\tau_l^2 v_{xo}^2}\right)R_{vac}^3\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^3.$$

$$R_2=R_{vac}\left[1-\left(\frac{2}{3\tau_l v_{xo}}\right)R_{vac}\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^2-\left(\frac{1}{2\tau_l^2 v_{xo}^2}\right)R_{vac}^2\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^3\right].$$

$$R_2=R_{vac}\left[1-\left(\frac{2}{3\tau_l v_{xo}}\right)\left(\frac{2v_{xo}v_{yo}}{g}\right)\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^2-\left(\frac{1}{2\tau_l^2 v_{xo}^2}\right)\left(\frac{2v_{xo}v_{yo}}{g}\right)^2\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^3\right].$$

$$R_2=R_{vac}\left[1-\frac{4v_{yo}}{3v_{ter,l}}\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^2-\frac{2v_{yo}^2}{v_{ter,l}^2}\left(1-\frac{4v_{yo}}{3v_{ter,l}}\right)^3\right].$$

Keeping terms of order $\left(1/v_{ter,l}\right)^2$:

$$R_2\approx R_{vac}\left[1-\frac{4v_{yo}}{3v_{ter,l}}+\frac{4v_{yo}}{3v_{ter,l}}\left(\frac{8v_{yo}}{3v_{ter,l}}\right)-\frac{2v_{yo}^2}{v_{ter,l}^2}\right].$$

Combine the last two terms to get:

$$R_2=R_{vac}\left[1-\frac{4}{3}\left(\frac{v_{yo}}{v_{ter,l}}\right)+\frac{14}{9}\left(\frac{v_{yo}}{v_{ter,l}}\right)^2\right].$$

Computer Exercise: My results are in "trajectories.xls"

Next two classes:

- Monday – Quadratic Air Resistance
- Wednesday – start Ch. 3

Code for (a)

```
from __future__ import division
from visual import *
from visual.graph import *

# Constants
g=9.8
b=0.1
deltab=0.05
tmax = 10
dt=0.001
rinit = vector(0,0,0)

# Objects
ball=sphere(pos = rinit, p=vector(8*cos(50*(pi/180)),8*sin(50*(pi/180)),0), m=1, radius = 0.1, color=color.red)
                # the above line defines a ball with initial position (10,0,0)m,
                # initial momentum (0,0,0)kg m/s, a radius of 0.1 m, and a mass of 1 kg.
vo = ball.p/ball.m
trail=curve(color = ball.color)     # This sets up the program for creating a curve which we will,
                        # inside the loop below, choose to trace the path of the ball
trail.append(pos=ball.pos)  # Start's tracing the ball's trail at it's initial position

print "b Computational range R(m) Drag-free range Ro(m) First-order range R1(m) Second-order range R2 (m)"
while b <2:
   t=0
   T = ball.m/b
   vter = g*T

   #Execution
   while not (ball.y<0):            # Observe that the loops is reentered only as long as t<tmax,
                    # so final time at which it enters will be between tmax-dt and tmax.
      F= ball.m*vector(0,-g,0)-b*ball.p/ball.m    # Calculate Force using the current v
      ball.p = ball.p + F*dt                    # Use the force to update the momentum
      ball.pos = ball.pos + (ball.p/ball.m)*dt    # Use the updated momentum to update the position
      t=t+dt                            # Advance the time by a step dt
      trail.append(pos=ball.pos)
```

```
                    # This sets the rate with which the loop is executed;
                        # to calculate as quickly as possible, comment it out,
                        # but to slow down the simulation, decrease the rate value

    b=b+deltab
    ball.pos = rinit
    ball.p = vo/ball.m

Code for (b) & (c)
from __future__ import division
from visual import *
from visual.graph import *

# Constants
g=9.8
b=0.1
deltab=0.05
tmax = 10
dt=0.001
rinit = vector(0,0,0)

# Objects
ball=sphere(pos = rinit, p=vector(8*cos(50*(pi/180)),8*sin(50*(pi/180)),0), m=1, radius = 0.1, color=color.red)
                # the above line defines a ball with initial position (10,0,0)m,
                # initial momentum (0,0,0)kg m/s, a radius of 0.1 m, and a mass of 1 kg.
vo = ball.p/ball.m
trail=curve(color = ball.color)     # This sets up the program for creating a curve which we will,
                        # inside the loop below, choose to trace the path of the ball
trailanal=curve(color = color.cyan)
trail.append(pos=ball.pos)  # Start's tracing the ball's trail at it's initial position
trailanal.append(pos = ball.pos)

print "b Computational range R(m) Drag-free range Ro(m) First-order range R1(m) Second-order range R2 (m)"
while b <2:
    t=0
    T = ball.m/b
    vter = g*T

    #Execution
    while not (ball.y<0):           # Observe that the loops is reentered only as long as t<tmax,
```

```
                # so final time at which it enters will be between tmax-dt and tmax.
    F= ball.m*vector(0,-g,0)-b*ball.p/ball.m    # Calculate Force using the current v
    ball.p = ball.p + F*dt                      # Use the force to update the momentum
    ball.pos = ball.pos + (ball.p/ball.m)*dt    # Use the updated momentum to update the position
    t=t+dt                                      # Advance the time by a step dt
    y = ((vo.y+vter)/vo.x)*ball.x + vter*T*log(1-ball.x/(vo.x*T))
    trailanal.append(pos=vector(ball.x,y,ball.z))
    trail.append(pos=ball.pos)
                        # This sets the rate with which the loop is executed;
                            # to calculate as quickly as possible, comment it out,
                            # but to slow down the simulation, decrease the rate value
Ro=2*vo.x*vo.y/g
R1 = Ro*(1-4*vo.y/(3*vter))
R2 = Ro*(1-4*vo.y/(3*vter)+(14/3)*(vo.y/vter)**2)

print b, "    ",ball.x, "        ", Ro, "       ", R1,"       ",R2

b=b+deltab
ball.pos = rinit
ball.p = vo/ball.m
```