Physics 1401V: Honors Physics I
TA/Instructor Manual

Fall 2014

An Important Note for TAs of 1401/1501

Until very recently, the PHYS1401 and 1402 labs were run in an *ad hoc* manner, with much left to the whims of TA's, instructors, and students. Over the last several years, the lab has slowly converged towards a more conventional style, in which each of the groups does the same exercise in a given week. Some of the exercises are "borrowed" from 1301, and some of them are unique to this course. The new "system" is much easier for the departmental lab supervisor to administrate, and allows for virtually any TA to walk in and teach a lab.

The labs are, on the whole, much more demanding than 1301, and the student lab manual is written in a very different style. Unlike the 1301 labs, there are numerous points (such as in the air resistance lab) where a reasonable amount of care is required to obtain good data and analyze it. The good thing about honors is that there will always be students who can meet your standard, no matter high how you set it. At the same time, I have learned over the years that students are going to pack up and leave at the end of two hours, if for no other reason than they have another class.

Students cannot afford wasted time because a TA was not prepared.

1. You must complete the lab yourself each week before the students walk in the room. If there is a problem, then work with Sean Albiston to fix it. He will always be able to set up the lab on Thursday afternoon of the previous week. This gives you ample time to do the lab.

2. You must also complete the analysis for the lab. Note that you are not merely expected to know how the apparatus works. You must know how to execute the experiment in the best possible way. This means doing the analysis, and if necessary repeating the experiment until you figure out how to do it correctly.

3. You must test all of the set-ups (at each table) each week. Once again, if there is something missing or some other problem, work with Sean Albiston to fix it. Arrive early for your lab period and make sure all of the hardware is working and is configured correctly. Understand the pitfalls and remind the students if there are common problems. **In past years, I have seen an extraordinary amount of time get wasted because TA's did not inform the students how to set up the Vernier hardware properly. This is your responsibility, as in many cases full instructions are not in the lab manual. The usual pitfall is the data collection rate.**

4. If something bad happens during the lab: a) find Sean to make sure it is fixed, and b) check in with the other TA's. Sean has spare equipment and a

lot of experience. Chances are that he can fix your problem in 5 minutes, but only if he knows about it.

Basic schedule:

Very tentative, and easily adapted
Week 1: no lab
Week 2: Lab 1.2
Week 3: Lab 1.2
Week 4: Lab 1.3
Week 5: Lab 2.1 and 2.2
Week 6: Lab 2.3 and 2.4
Week 7: Paper 1
Week 8: Lab 3.1 and 3.2
Week 9: Lab 3.3 and 3.4, Rewrite 1
Week 10: Lab 4.1
Week 11: Lab 4.2
Week 13: Thanksgiving
Week 14: Lab 5.1
Week 15: Lab 5.2
Week 16: Paper 2, return at finals

Before you get started with labs…

1. Get yourself a copy of 1301/1302 TA manuals. You can ask for these at the front desk of the Physics building, or ask the lab coordinator, Sean Albiston. If you get them from the front office, have the professor teaching this course send along permission for the hard copies. Electronic copies and other resources are available at the following website:

http://groups.physics.umn.edu/physed/Research/Lab%20Manuals/Lab%20Manuals.html

We share quite a bit of material and equipment with these courses and the manuals have many useful tips on using it, along with some data. The appendices of the student course manuals are also quite helpful for these purposes*.* *However,* there should not be any **vital** information missing from this guide, if you feel this guide is lacking important information needed to perform a lab, please let us know.

2. We also use the same video acquisition LabView® program, VideoRecorder, as the 1301 class. For the anlaysis, we use a freeware program called *Tracker*. You can find information and documentation at the following site:

http://www.cabrillo.edu/~dbrown/tracker/

Tracker does have its limitations, so, in conjunction, we use Microsoft® Excel for the actual data analysis. The data can be easily copied from *Tracker* in to a spreadsheet. Become familiar with this program.

3. Another piece of software that we use in both 1401/1501 and 1402/1502 is Vernier's Logger Pro®. This is used in the regular 1202 and 1302 classes, but we've found use for it in the first semester as well, primarily for photogates and sonic motion detectors. If you are not familiar with this program, there are instruction manuals available online.

In 1401/1501, we use the Vernier photogates and motion detectors. You must make sure these are working properly before each lab period. Most of the problems that occur with these have to do with software settings and not failure of the hardware.

Table of Contents

Lab 1: Kinematics

Lab 1.1: Galileo's Experiment: Measuring Acceleration Due to Gravity

Purpose:

In this lab students will drop similarly sized balls (of various masses) and record the vertical fall of each using a video camera. They will analyze the video and fit the trajectory with a parabola that will yield each ball's acceleration, which is approximately **g**, 9.806 m/s^2.

The focus of this lab should be on how the students relate the motion of the falling ball with the parabolic plot of its position as a function of time. Also, as with all labs in this course, students should also be paying close attention to accuracy, precision and uncertainty of their measurements, and how these relate to their equipment and procedure.

Materials:

Students will need the following materials:

- a meter stick (and a ring stand - optional),
- tape,
- 1 set of 1″ (2.5 cm) diameter balls
  - tungsten carbide (~128 g)
  - steel (~66 g)
  - aluminum (~23 g)
  - white plastic (~11 g)
  - polyethylene plastic (~8.8 g)
  - wood (~4.9-6.0 g)
  - hollow plastic (~2.6 g)
- bubble wrap and tray,
- a video camera and tripod, and
- a computer with VideoRecorder and *Tracker* programs.

Expect the students to use as many of the different balls as time permits for this first experiment.

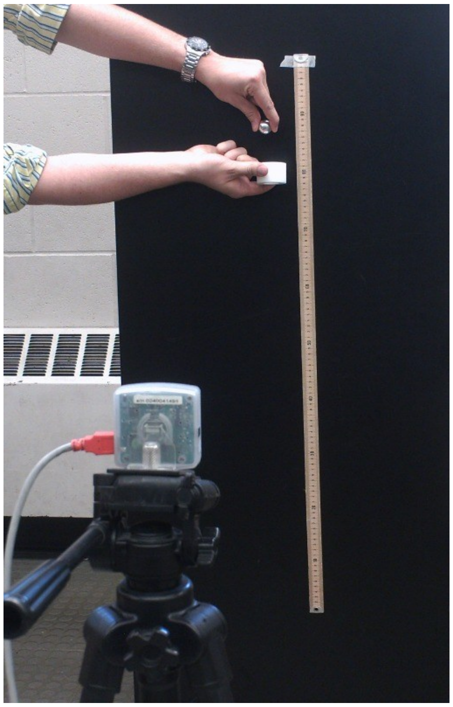
The masses of the balls vary from the listed values, make sure student confirm the mass of each ball they use.

Use bubble wrap in a tray to catch the balls. **Do not let them hit the ground**. Damaging the balls will impact the results for the next lab.

Setup:

This lab is essentially the same Lab 2, Problem 1 in the Physics 1301 lab manual. This may be a good resource for preparing for lab or leading discussions.

Make sure that the cameras are set up properly: i.e. short exposure time, large gain. Make sure you know how to adjust all of the camera settings and make sure that your students have set it up correctly. You can focus the camera by twisting the silver lens ring.



Tape the meter stick to a surface so that it is completely vertical (the ring-stands work well for this). This will be used for calibrating the length per pixel in the video analysis program. Aim the camera on the tripod so that its line of sight is horizontal and centered on the 50 cm mark of the stick. Also make sure that the camera is focused properly on the ruler and ball. Try to get the entire meter stick in view. You should remind your students that a longer range will produce the best results.

Procedure:

Hold the one of the balls in the plane of the meter stick in the view of the camera. Capture the fall of the ball using the LabView® program, VideoRecorder. Once you have recorded a good video, save it to the desktop of your computer.

Because the balls do tend to stick to your during the release, you may use the dropping-mechanism to consistently drop the ball as vertically as possible; however, using this mechanism means that you cannot capture the position of the ball just as it is released. You will have to account for this in your analysis. Keep in mind that it is important that the ball fall in the same plane as the meter-stick relative to the camera. If not, then the calibration will not represent the actual distances the ball has moved.

Repeat this process for each of the balls.

Data and analysis: **Using *Tracker***

Analysis of the video is similar to that which is done in the 1101, 1201 and 1301 classes, except that you will be using the *Tracker* program instead of LabView®. The advantages to the *Tracker* program are that you can always go back and recalibrate or re-enter data points whenever you want. There are no predictions to enter in, and the data easily exports to a Microsoft® Excel file. The disadvantage is that it is less straight forward and you can skip important steps easily. Important features are:

1. Calibration tool: sixth button from the left on the tool bar. It looks like two '+' signs connected by a diagonal line with a 10 next to them. This tool lets you draw a line and tell Tracker what real world length it has. Clicking the button will give you the option 'new' from which you can choose what type of tool. Use either the stick or the tape, they both let you set the calibration of the world.
    a. Measuring stick sets a hard calibration value. Once you type in the value, changing the length of the stick changes the calibration of the video. This is likely the best tool to use for calibration.
    b. Measuring tape will "measure" the distance that it spans. Changing the length of the tape will change the number it displays. Thus if you have already set the calibration, you can use it to measure distances on the video easily. You can use it to set calibration by clicking the number box and changing the value.
2. Coordinate system: seventh button from the left, looks like a purple set of axis. Click and drag the origin to where you want it. You can rotate the axis by click on the horizontal dash mark on the x-axis. By dragging the hash mark along the axis you can get more precision over how much the axis rotates. **Students should have a perfectly vertical object so they can set the axis to be aligned with it.**
3. * Create: Eight button from the left. Click it and choose point mass. This creates a graph in the upper right corner and a table in the lower right corner. You can change which variables this program looks at by clicking the axis of the graph or the 'table' button in the table. It defaults to showing the x vs t graph and t,x,y on the table.
    a. The graph is useful mostly for visualizing data as it happens. You can right click and select analyze to do fitting (auto fit and by hand) but it can be a very frustrating tool to do so. In general we advise you use excel for analyzing the data.
    b. The table can be highlighted and copy and pasted into excel. When you paste it, it always includes the name of the point mass (usually mass A) and the variable headings in the first two rows of copied data.

Students should export the data to Excel and use Excel's fitting function to determine the acceleration of each ball.

They should also note that there may be some random variance in the horizontal direction due to the placement of the cursor, (if the ball had fallen straight down). Make sure that the students are aware of this random error. If the ball had not fallen straight down, they should notice a general trend in the horizontal direction.

Make a plot of the acceleration against the mass of each ball. The results should reasonably show that the effective acceleration of each ball is inversely proportion to its mass.

Notes and Comments:

You may notice for this lab that the parabolas of the position versus time graphs are curved a bit more than they should be. This is mostly due to a wide-angle distortion from the camera lens. The effect is not very important for the 1201 or 1301 labs, but since we are focusing on accuracy and careful measurement, this can affect students' measurements (you can even see the effect on an object moving with constant velocity).

One question that tends to arise is: "Where on the ball should you measure from?" The key here is to be consistent. If the ball is moving downward, use the very bottom of the ball. You know that there should be 1/30 of a second between each frame, so there must be 1/30 of a second between each time that the shutter closes. Using the center of the ball will not be as accurate because the blurred images will not give you a clear position to measure from.

You will not have too many data points, and the best ones will be with the ball moving slowly. With that in mind, the lab could be done by throwing ball upward and capturing the top part of the parabola. A variation of this technique involves playing "catch" with a ball and filming the flight. Note, though, that it is more difficult to keep the ball in the plane of the ruler using this method.

IMPORTANT NOTE:

This lab is meant to be done by hand, so that they get an idea of how to fit lines to a chart. They should calculate the velocity as instructed in the manual and then fit that plot "by hand". If you simply use the "trendline" option you will get bad results (gravity will usually be well above 10 m/s^2). This is because at such short distances, the difference between g=9.81 and g=10.5 is very small, well within the uncertainty of the plot.

When students do the hand fitting, they should find they can get a slope of 9.81 to match, but make sure they explore the limits; how low can they still reasonably match, how high can they reasonably match? Your goals should be:

1. Get them familiar with data taking and analysis.

2. Get them to see the uncertainty of the slope and intercept by having them make several fits for the line. If they are printing out the plots, have them draw the uncertainty bars on the plots. They may
3. THE R^2 VALUE IS MEANINGLESS TO PHYSICS! It is a measure of correlation, and all measurements we make will be highly correlated even if the data is very poor. DO NOT ACCEPT STUEDENTS QUOTING YOU THE R^2 VALUE!
4. Realize the limitations of this method of measuring gravity. You need much longer distances (perhaps 4 meters or more) to get decent results given the amount of uncertainty we have.

Have them obtain the result for 'g', with the uncertainty, for each plot.

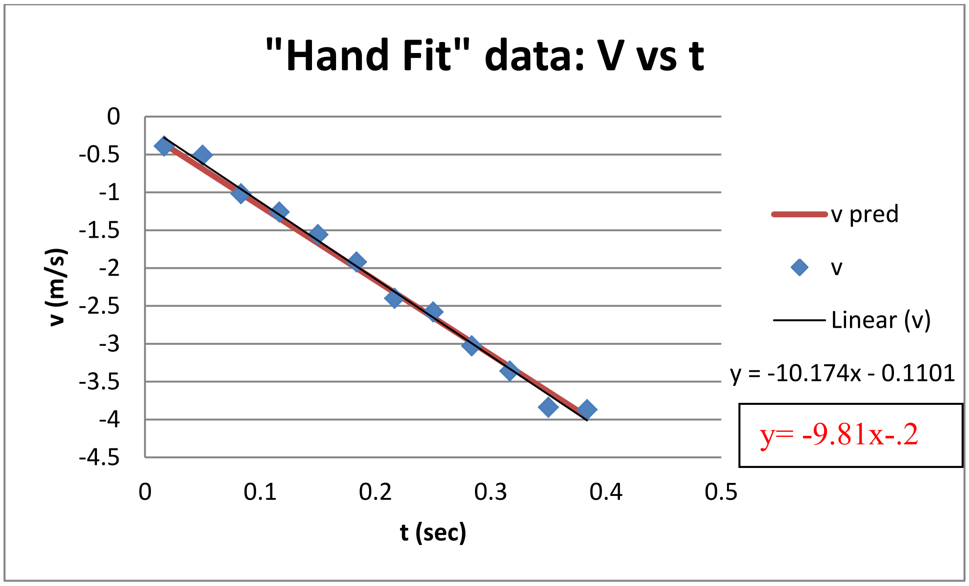


Figure 1: Comparison of trend line fit vs "correct" fit (in red). The two are practically indistinguishable, which should help students realize the limits of this method for measuring gravity. Despite having an R-value of .9938 the result isn't 99.38% accurate, which students may try to tell you. A reasonable reporting might be g=-10.2±.4 m/s^2, v0=-0.1±0.1 m/s (though the uncertainties are likely even larger).

Note on R^2: this value is only useful for sociology or an experiment where no correlation is known to exist. It does not comment on the uncertainty of the slope or intercept in any meaningful way. They may argue that a low R-value will tell you if the data is inconsistent, but at that point it will be visually obvious as well. They don't need this number. **Ever.**

Lab 1.2: Gravity and Air Resistance

Purpose:

This lab is a more precise version of the free fall with camera. Students use photogates to time the passage of a dropped ball. The focus here should be on careful setup and measurement.

Materials:

The materials needed for this lab are the following:

- 1 Pasco® aluminum track (2 m in length),
- a c-clamp and block of wood,
- 2 Vernier photogates with thumb-screws and square nut,
- a Vernier SensorDAQ® or LabPro® interface,
- a platform with dropping mechanism and plumb-bob,
- 1 set of 1" diameter balls
  - tungsten carbide (~128 g)
  - steel (~66 g)
  - aluminum (~23 g)
  - white plastic (~11 g)
  - polyethylene plastic (~8.8 g)
  - wood (~4.9-6.0 g)
  - hollow plastic (~2.6 g)
- basket with bubble wrap
- a computer with Vernier Logger Pro® software tool.

The masses of the balls vary from the listed values, make sure student confirm the mass of each ball they use.

Before the students arrive:

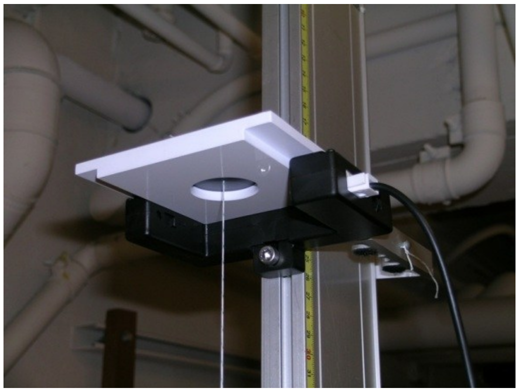
- Make sure the 2m tracks are properly clamped to the tables with the bottom end against the floor.

- Make sure that all of the balls fall neatly through the dropping jig. When we were using sand, it tended to get stuck inside the cylinder. That problem should be gone. They should not need to push the ball through the cylinder.

- Install both photogates and make sure that they are attached to the tracks **using the correct hardware**. Note that the students will have to tighten these so that they do not move. **A ball should be able to hit the bottom photogate without it moving**.

- Make sure that the photogates are working and that they are properly set up on the computer.

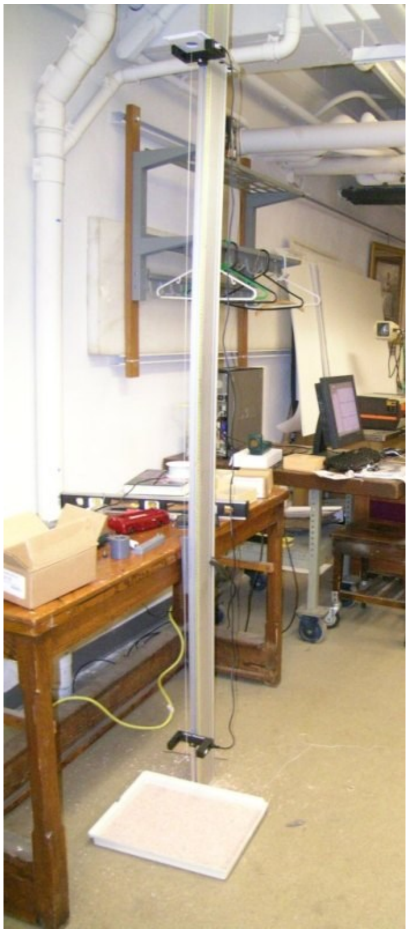
- Install the jig and plumb bob in the top photogate. You can leave the alignment for the students.
- Make sure the bubble wrap and tray are in place.



The dropping platform/jig (right) was made specifically for this class and for this lab. The platforms are likely permanently attached to the photogates. If not, then simply slide the platform onto the photogate. Double-sided tape should be used to keep the platform from moving during the experiment.

A plumb-bob and cylinder were made to fit into the platform. The plumb-bob will be used to align the photogates in the setup. The diameter of the cylinder is just over 1”. When aligned correctly, this will produce drops that do not miss the gates and are repeatable to about 1 millisecond if your gates are over 1 meter apart.

See the note above about the cylinder, which is essential for getting reproducible drops!



The photogates have two modes: using internal and external light sources. Only use the internal source. You can switch to this mode by unblocking the detector on the inside of the photogate (they should be in this state already).

Setup:

To begin the setup, mount the track vertically by clamping it to the edge of a table with a c-clamp and block of wood, as shown in the figure to the left. The track should stay in place when the bottom is resting on the floor. This also allows for easy adjustments to the alignment. Place the tray with bubble wrap at the base of the track.

Mount the photogates on the sides of the track with the nuts. Make sure that the gates are attached are very tightly, so that they do not rotate when a ball hits them. Set the photogates as far apart as possible (1.7-1.9 m), and record this separation as accurately as possible. Make sure that the students measure the distance correctly! With the dropping mechanism available, the lab should be done with only two photogates. The dropping mechanism should be attached to the upper photogate (see the figure on the previous page).

The major challenge here is properly setting up and aligning the gates. **Note that this must all be done very carefully to observe the effects of air resistance. For example, for a fixed drag coefficient (using balls of identical diameter), a solid plastic ball will take about 3 milliseconds more to fall 1.5 meters than a steel ball. This type of precision can be achieved with care. This is an excellent lab**. Lean on the students to do it well and they will learn a lot.

Place the plumb-bob into the platform on the top gate. Use this to align the track so that it is vertical and to ensure that the ball will fall directly through the two gates. Both gates should be level.

The alignment of the top photogate is critical, because in order to get the initial velocity correctly, you need to make sure that the center of the ball passes through the top gate (and then the velocity is the diameter of the ball over Δt). The bottom photogate is less critical; just make sure that the plumb-bob passes through the light beam. When aligned correctly, the cylinder will produce drops that do not miss the gates and are repeatable to about 1 msec if your gates are over 1.5 meters apart.

Connect the gates in a daisy-chain mode: each gate connects into the other, and the last one into the Vernier Logger Pro® interface (as shown left). Click on the "Experiment" menu in the Logger Pro® window, and then select "Data Collection." Check the option for "Continuous data collection" and **adjust the sample rate to somewhere around 3000 samples per second (~5000 is the maximum sample rate). If the rate is not set properly, this experiment will not work.**

Procedure:

Use the set of the 1" balls for this lab. These start with the (steel) balls and end with the lightest (hollow plastic) balls. There are also tungsten carbide balls, but these have to be shared. Note that dented balls will have less repeatable drops, so be careful not to let the balls stray and hit the photogates or other equipment. The rubber ball has a noticeable seam and is less repeatable than the others.

Press the "Collect Data" button at the top right of the Logger Pro® window. Carefully drop the balls through the dropping mechanism into the tray below to avoid any bounce. A good drop will result in 4 recorded times (two from each gate – when the ball enters and leaves each gate). Copy and paste the time data into an Excel file.

Repeat this process for all of the different balls.

Data and Analysis:

If you think this through, it is all a matter of determining an effective g from the usual formula:

$$\Delta y = v_i t + \frac{1}{2} g t^2 .$$

Let t_1 and t_2 be the two times from the top photogate. Then

$$v_i = d/(t_2 - t_1),$$

and the drop time *t* is the difference between the two times when the ball is at the center of each gate:

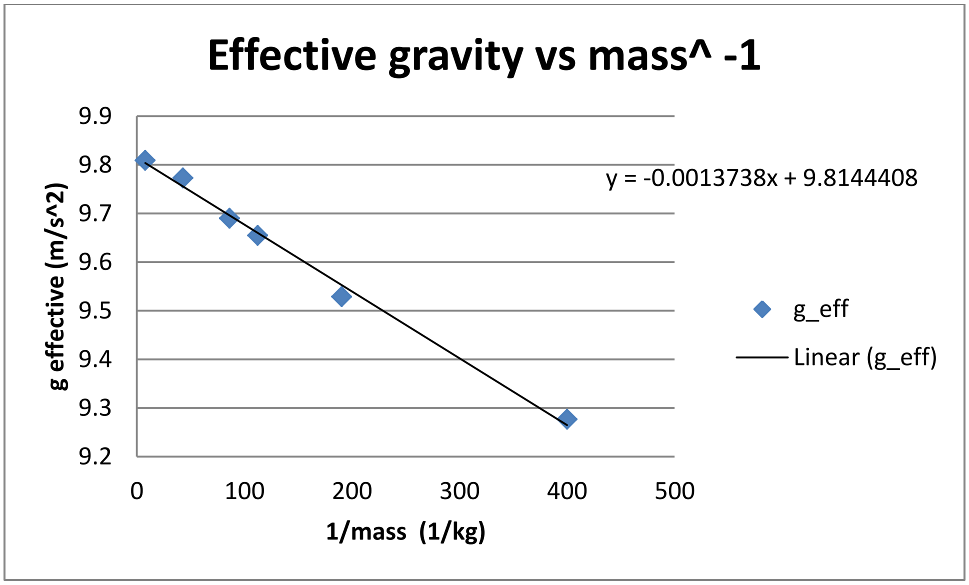
$$t = \frac{(t_3 + t_4)}{2} - \frac{(t_1 + t_2)}{2} .$$

The students can measure Δy precisely using the scale on the edge of the tracks. It is of course very difficult to determine exactly where the light beam is, but both gates are identical in size, and so they can just measure the distance between the top or bottom edges of the two gates. Then

$$g_{eff} = 2\frac{(\Delta y - v_i t)}{t^2} .$$

$$g_{eff} = g\left(1 - \frac{kY}{3m}\right).$$

You should plot the effective **g** of each ball as a function of its inverse mass (plotting it vs mass directly will be quite hard to fit. If you do plot it this way, you should see an asymptote near 9.81). A plot of the effective acceleration vs. one over the mass should give a straight line with a negative slope (kY/m) and an intercept near 9.81. The fit will not be perfect, since there were some assumptions made, like those in equation 1.12 of the lab manual.



The drag coefficient obtained from this data is approximately C=0.7 (most sources place C for a sphere around 0.5±0.3).

Notes and Comments:

The heaviest balls are the best for measuring **g** itself, and the students should start with those. You should be able to get within 1% of the correct g with the steel and tungsten balls. The lightest balls will give an effective g that is about 6 – 7% too small. The students can plot the effective *g* as a function of the mass of the balls.

The second thing that students can look at in this lab is the prefactor in the inertial drag term, assuming that the drag is quadratic in velocity. This can be obtained from the slope of the line when g_{eff} is plotted vs. *1/m*. Using the slope they can calculate the 'C' (drag coefficient) of the balls. Expect students values around 0.5±0.3 (sources vary on what C 'should' be, and it varies base on the dynamics of the ball, but C=0.5 is a common value stated).

The students might be tempted to determine g_{eff} using the kinematic equation $v_f^2 - v_i^2 = 2ad$. They should rapidly find that this approach will not work, because it is virtually impossible to ensure that the ball passes through the lower photogate on a line passing through its diameter.

This lab is VERY sensitive to small fluctuations in values. An entered total distance of 1.93 meters and 1.933 meters will have noticeable effects on calculations. The diameter of the ball is like-wise sensitive. Should students find they are getting values around 9.82+ for the heavier balls, it is usually because they included more (or less) digits than they should have. Rounding down the total distance and adding more precision to the ball's diameter can help "fix" a g_eff that is too high.

Lab 1.3: Projectile Motion in Two Dimensions

Equipment

- Pasco **short range** projectile launcher with plastic balls
- 1 photogate
- 1 ring stand
- Vernier LabPro computer interface
- 1 desktop computer
- 2 wooden blocks
- Cork board wrapped in a fresh (unwrinkled) piece of aluminum foil.
- measuring tape (Make sure that Sean provides you with enough of these. They are the long tape measures on a wheel.)
- 1 table clamp
- 1 stopwatch

This should be a relatively easy lab for your students. After they determine the muzzle velocity (which does require the computer), they may disperse into the hall to do the rest of the lab, or they can do it all in the lab (which can be a bit chaotic, so if you do this stress safety). The maximum range will be about 2.5 meters

It is essentially a matter of verifying the range equation. Students will need to use the version that does not assumed y0 =0, since the launcher stand adds ~24 cm of height.

$$x = \frac{v_0 \cos\theta}{g}\left(v_0 \sin\theta + \sqrt{v_0^2 \sin^2\theta + 2gy_0}\right)$$

If y0=0, $x = \frac{v_0^2 \sin^2\theta}{g}$, which students will try to use if you let them. An important feature of a non-zero y0 is that the max distance is no longer at 45 degrees. Students can predict the angle that leads to maximum distance (either by plugging in values of using derivatives). It should be a bit less than 45, about 40 if the only height is from the stand. Have them take extra data points in this region to drive home the point.

They set up the cannon, position the cork board, and then measure where the ball lands. The velocities are small enough that air resistance is not so important. They do this for several angles and plot the range vs. angle.

Muzzle velocity varies with ball mass and angle of launch. If different masses of balls are used, make sure students find the velocity for each

DO NOT USE METAL BALLS. That would be dangerous. Using the velocity at 45 degrees is a good average. If done in the lab near a computer, students can also set the photogate to measure velocity with each trial.

You can expand this lab by having them fire from different heights, from a table to the floor, from the floor to a table. Floor to table is interesting because there is a minimum angle for a ball to reach the height:

$$if\ y_0 < 0,\ \ \theta_{min} > \sin^{-1}\sqrt{\frac{2g|y_0|}{v_0^2}}$$

PRO-TIP: If you use excel, the SIN(), COS(), functions use radians. You can do SIN(RADIANS(number)) to quickly convert to radians and perform sin/cos on a number.

For best data:

1. Make sure students are accurate in their measurements/placement of x0 and y0.
2. Make sure the photogates are centered properly so the v0 value is correct (being off-center will raise v0). **This is one of the most common problems in this lab.**
3. Make sure that they are accurately measuring the angle.

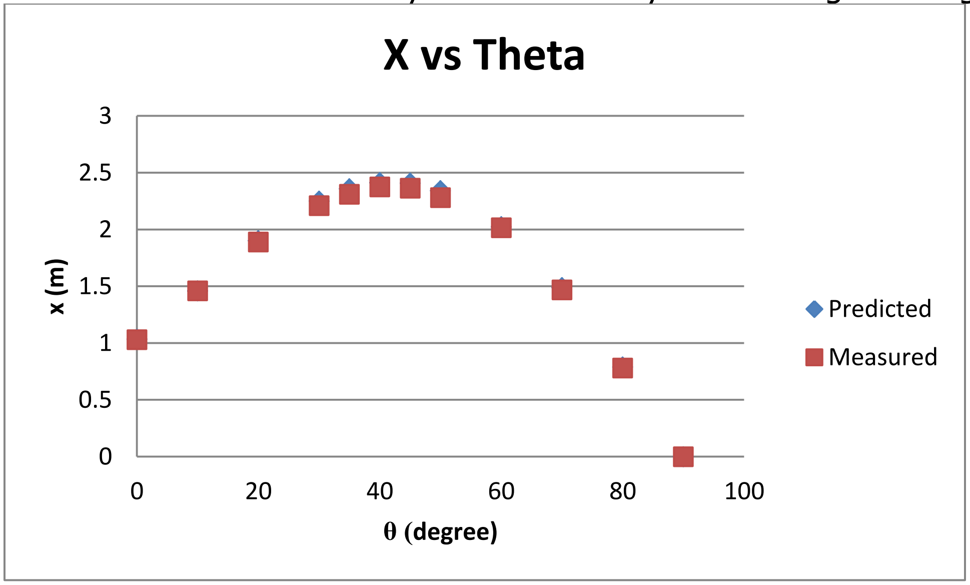


Figure 2: typical results for y0=24.5 cm, v0=4.63 m/s using the white plastic balls

Lab 2: Momentum and Energy

Lab 2.1: 1-D Collisions: Colliding Carts

Purpose:

This is a classic 1-D collision lab. Students will confirm the conservation of momentum in 1 dimension as precisely as possible using the Vernier photogates. This lab could be very instructive in building intuition about the final motion of the carts based on their masses.

For this lab, we intend to use the photogates for measurement, since they are much more accurate. The lab could be done with cameras and video-analysis software, instead, but we find that more time consuming and the parallax effect to be too annoying to deal with.

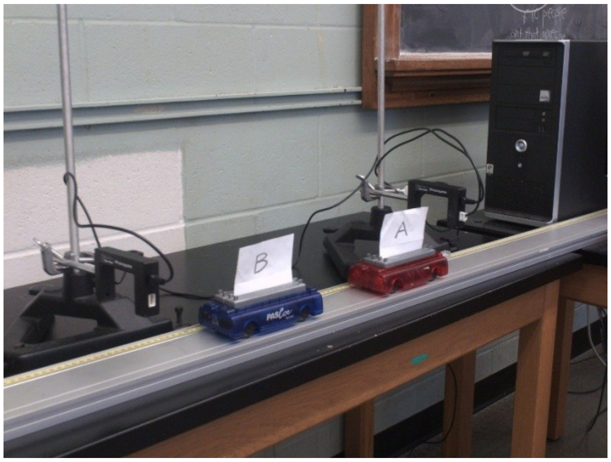
Materials:

The following materials may be used for this lab:

- a Pasco® aluminum track (2 m in length),
- 2 Pasco® carts (Velcro® and magnetic sides, ~250 grams each),
- about 4 Pasco® 250 g bricks for the carts,
- 2 Vernier photogates,
- a Vernier SensorDAQ® or LabPro® interface,
- 2 ring stands and clamps,
- a balance,
- a meter stick,
- some scrap paper,
- tape, and
- a computer with the Vernier Logger Pro® program.

Make sure that you have only "good carts!" Bad bearings = Bad data. Sean keeps a nearly infinite supply of new carts. Make sure any bad cart is REMOVED from the lab and replaced!

Setup & Procedure:



There are a number of things that can be done with this lab, but it may be most useful to determine how the final motions of the carts after a collision are affected by each cart's mass (essentially the same as lab 6, problem 1 in the 1301 lab manual). However, you should encourage the students to explore different collisions, whether elastic or inelastic.

To begin, set the track on a level surface. Next to the track, place the two ring stands, each about 1/3 of the track's length from each end. Clamp one photogate onto each of the ring stands such that the infrared beam of the gate is horizontal (as shown to the right).

Tape a square piece of paper with well-cut edges to the side of each cart so that the paper stands vertically. This paper will be used to pass through and trigger the photogates. The paper should be square, and the length of the paper measured so that you can measure the velocity of the carts.

For elastic collisions, position the carts so that both their magnetic sides will collide together. For inelastic collisions, have the Velcro® sides face each other.

With the and the Logger Pro® program open on the desktop, open the "Experiment" menu, and select "Data collection." Check the option for "Continuous data collection" and set the sample rate to about 1000 samples per second.

Have one cart begin at rest between the two photogates (cart "B" in the diagram above). Press the "Collect" button at the top right of the Logger Pro® window. Launch or push the other cart (cart "A") from the end of the track toward the stationary cart ("B"). This moving cart should pass through one of the photogates before the collision. After the collision both carts should pass through one of the photogates. Which of the photogates it passes through depends of the masses of the carts and whether the collision is elastic or inelastic. Pay attention to when each cart passes through a gate so that you can double-check whether times were recorded properly.

For best results:

1. Use higher mass and higher velocity. If the velocity is too low, you will notice the slowing of a cart due to friction, even though it is low it will be a noticeable % at low speeds. At higher speeds you get much better results. If you use a cart with no masses in it, it will often jump off of the track (especially at higher speeds), so put at least 250 grams of weight in it to begin with.
2. Head on collisions don't work well. They cause a lot of vibrations, so you end up losing energy. They also often end with very low momentum so they suffer from previously mentioned problems.
3. Make sure students decide on which direction is 'positive' momentum, and to note when a cart is going in the 'negative' direction. The photogates will not 'automatically' tell you what direction a cart is moving in.
4. Make sure tracks are level and carts don't move on their own.
5. Some runs are just better than others, if data is really bad, retry it.

Data and Analysis:

Divide the length of the paper on the carts by the times it took for each to pass through the gates. This will be the carts' speeds just before and after the collision. Multiply these values by the respective masses to get the momenta.

In each of the cases, check to be sure that the total momentum before and after the collision is conserved. You and students may notice that there is some friction, and so a minimal speed which will produce acceptable results. The goal is to get it within 5%, which is feasible.

You should also be able to predict what the final momenta will be based solely on the masses (*m_1* and *m_2*) and initial velocities (v_{1i} and v_{2i}) of the carts – we are assuming that cart 2 is initially at rest.

For inelastic collisions, you only need the conservation of momentum – kinetic energy is not conserved in these collisions. The final velocity, v_f, is

$$v_f = \frac{m_1}{m_1 + m_2} v_{1i} .$$

For inelastic collisions you can also predict the percent energy lost:

$$\frac{\Delta E}{E} = \frac{m_1}{m_1 + m_2}$$

For elastic collisions, the equations become a little bit more complicated, but you should find that the final velocity for the first cart, *v*1f, is

$$v_{1f} = \left(\frac{m_1 - m_2}{m_1 + m_2} \right) v_{1i}$$

and the velocity of the second cart, *v*2f, is

$$v_{1f} = \left(\frac{2m_1}{m_1 + m_2} \right) v_{1i} .$$

Note that the sign of the final velocity of the incoming cart is dependent on the masses of both carts and that it is zero when both carts have the same mass.

If energy conservation or efficiency is done, the coefficient of Friction can be estimated by assuming all 'missing' energy was lost due to friction:

$$\mu mgx = E_{lost}$$

The coefficient of friction should be low (on the order of 0.004-0.001 seemed to be the consensus of students, which is believably low).

Notes and Comments:

As noted above: replace any bad carts immediately.

Place the photogates as close to the impact zone as possible, but make sure that they fully leave the gates before colliding.

This experiment may also be done with the sonic motion detectors. The motion detectors have the advantage that they give second by second information about velocity, so you can pick the velocity just before and just after the collision. The downside is that motion detectors can be "jumpy" (their output fluctuates greatly) and can lose track of their target leading to problems.

Make sure they understand how the uncertainties propagate to the final results, the change in momentum will not be zero, but it will likely be within one or two uncertainties (δp_{tot}) (about 5-10% on initial and final momentum values).

$$\delta p_{tot} = \sqrt{\delta p_i^2 + \delta p_f^2} \approx \delta p_i + \delta p_f$$

$$\Delta p = 0 \pm \delta p_{tot}$$

Lab 2.2: Two-Dimensional Collisions

Purpose: To see the conservation of momentum in two dimensions.

Equipment:

- An air table
- Several plastic pucks, some with velcro, some without
- Video camera
- a desktop computer
- a meter stick
- a level
- shims or pieces of card board used for leveling the air table

Air tables need to be as level as possible, but they will never be perfect. The goal is to have the 'natural' accelaration as low as possible and keep the collisions fast enough that it doesn't really matter.

Use the table clamps and rods to mount the cameras levely, don't try and use the tripods. The camera need height over the table to see the motion, but to still be close enough to see details on the calibration object (about 80cm or so). Students need to make sure they have enough visibility of BOTH pucks to take data before and after the collision (although in my experiments, 3 data points still gave reasonable results).

The experiment conserves momentum about as good as the other collision labs, but has real problems with energy conservation; the rebound of the pucks is VERY inefficient and the ’elastic’ collisions loose about 50% energy. **Only use this lab for momentum, not energy.**

Inelastic collisions work well, though make sure students keeping spining after collision to a minimum or else they will not have constant final velocities.

Taking the data can be done pretty quickly, if the students remember tracker. They should use two masses points (mass A and mass B) using the ”Create *” button for the two different pucks. They should get time vs x and y for each puck for before and after the collisions. The students will need to analyze the data to find EIGHT velocities: vy and vx for two pucks, before and after the collision. This can be done by fitting x vs t plots, but can take a lot of time.

Make sure students take all the videos and use tracker to get all the data saved BEFORE they start to analyze the velocities. They can analyze velocities at home, they can’t take more data at home. Keep in mind that it may take them a decent amount of time out-side of class to analyze the data.

Make sure they understand how the uncertainties propagate to the final results, the change in momentum will not be zero, but it will likely be within one or two uncertainties (about 5-10% on initial and final momentum values).

$$\delta p_{tot} = \sqrt{\delta p_i^2 + \delta p_f^2} \approx \delta p_i + \delta p_f$$

$$\Delta p = 0 \pm \delta p_{tot}$$

	x-velocity (m/s)	y-velocity (m/s)	x-momentum (kg*m/s)	y-momentum (kg*m/s)
Puck A initial	-0.9293	0.0081	-0.0284	0.0002
Puck A final	0.3196	-0.0531	0.0097	-0.0016
Puck B initial	0.5106	-0.1553	0.0156	-0.0048
Puck B final	-0.6666	-0.0525	-0.0204	-0.0016
Total initial	--	--	-0.0128	-0.0045
Total final	--	--	-0.0106	-0.0032
Final-initial (Δp)	--	--	0.0022	0.0013
δp_{tot}			0.0020	0.0020

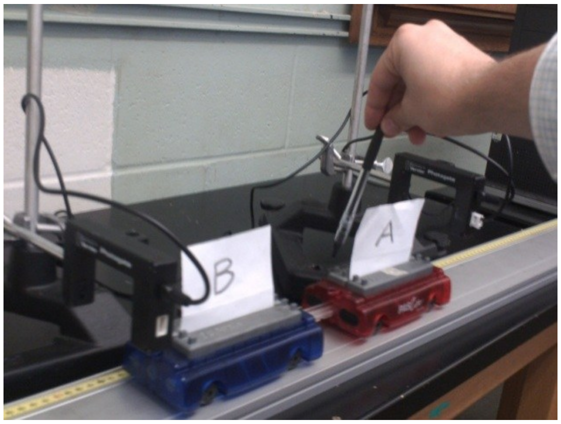
Example of data

Lab 2.3: Energy, Explosions, & Spring-Loaded Carts

Purpose:

Two carts start from rest in contact with each other. They will be launched apart like in a 1-D explosion (a total momentum of zero). This can be used either as momentum conservation or a spring potential-to-kinetic energy conversion lab.

Setup and procedure:



The materials and most of the setup for this lab are identical to the previous collision lab. The two carts start near each other, one of them with its spring-launcher loaded. Both carts can (and should) have various mass blocks on them. Again, having some initial mass added makes the carts much more stable.

Set the carts between the two photogates and begin collecting data with the Logger Pro® software. To launch the carts, carefully tap on the spring-release button on top of the one cart. What works best is striking it with the flat side of a pen in a careful but fast vertical motion. Using a finger may slow down the cart as it slides off.

Data and Analysis:

Now you can check that the total momentum (which should be zero) is conserved.

If your students are interested, they can look at the energy of the carts after the collisions. By using three different spring levels, you can compute the final kinetic energy of the carts, and check that indeed it goes as number of notches squared. This works pretty well. Additionally, one can check the spring constant by loading the spring with some weight and measuring the sag (keep the release button down). Note that the spring is compressed within the cart even when at full extension, thus their F vs X plots should not intercept zero.

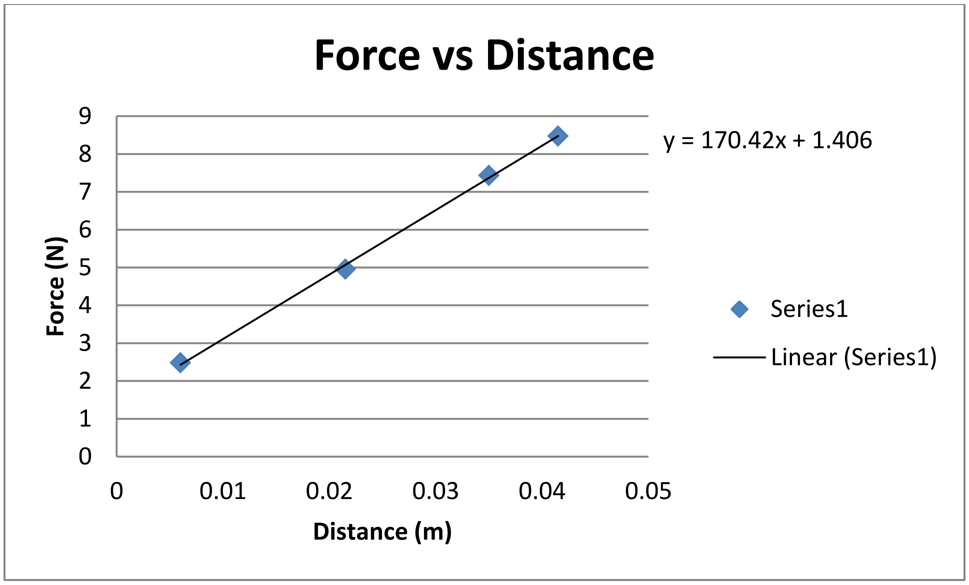


Figure 3: Determining the 'k' constant for the spring. Note that the intercept is non-zero because even at x=0 (full extension of the prong) the spring is still compressed in the cart a small amount.

Friction is noticeable in its effect on the total energy, so the students can again explore the work done by friction. So make sure they measure how far the carts travel before their velocity is measured. They should find that the coefficient of friction is around 0.004-0.001.

Notes and Comments

We have found that this is a good exercise to remind students that momentum is a vector. Many will automatically say that momentum is not conserved in the explosion because the carts began with no velocity, but then both ended in motion. However, the direction of each cart's motion is important, and it is good to remind the students of this. They should also know that the total momentum will not be affected because the "explosion" is an internal force.

Lab 2.4: Momentum and Energy: The Ballistic Pendulum

Purpose:

This conservation of momentum experiment is also meant to be very precise. Students should take careful measurements in order to verify the exit velocity of a ball from a launcher. This velocity will be measured with a photogate, and then derived from data on the pendulum.

Materials:

The following materials may be used for this lab:

- a Pasco® projectile launcher (long or short range),
- a Pasco® table-clamp or C-clamp,
- a steel 1" (2.5 cm) diameter ball,
- a Pasco® Ballistic pendulum set-up
-

Setup:

The current set-up is a self-contained ballistic pendulum unit. The ball is fired into a pendulum and then the pendulum swings to a maximum height. The ball catcher should have some brass weights on the bottom. A small black stick moves with the pendulum and measures the maximum angle it reaches. Steel or Tungsten carbide balls can be used.

Procedure:

Load a steel ball into the long-range projectile launcher.

Repeat the experiment by using a different setting on the launcher, or adding mass to the catcher.

Data and Analysis:

From the data from the photogate, determine the velocity of the projectile as it exits the launcher. With the video analysis software, you will determine the maximum vertical displacement of the catcher.

From the conservation of momentum and energy, you will be able to find that the final height of the catcher should be

$$h = L_{cm}(1-\cos\theta) = \frac{v_{ball}^2}{2g}\left(\frac{m_{ball}}{m_{ball}+m_{pendulum}}\right)^2$$

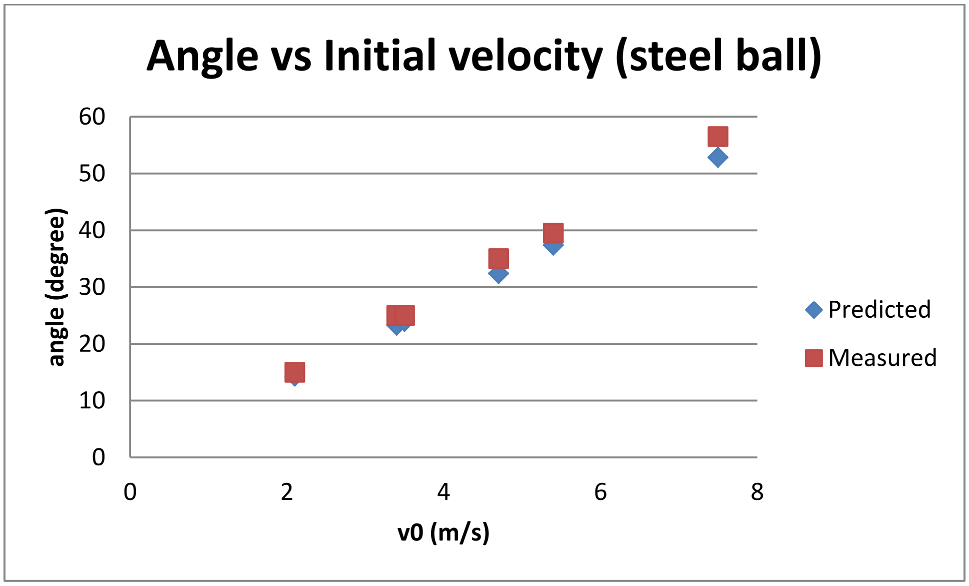
The length of the center of mass can be found by letting the pendulum (with ball inside) hang so it can swing freely (tilting the set-up forward). By measuring the period of the oscillations you can find out what the effective center of mass is:

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{L_{cm}}{g}}$$

The weight of the ball catcher plus the ball should cause the center of mass to be centered more or less on the center of the ball (about 30cm)

Notes:

1. A huge calculation error occurs if students use h=L*sin(angle) instead of h=L(1-cos(angle)).
2. Students sometimes ignore the collision and simply try to use the initial ball velocity for the KE.
3. Not correctly finding L from the center of mass. If they simply use the length of the aluminum rod or the bottom of the ball catcher, they will be introducing very noticeable error.
4. Comparing predicted angle to measured angle is more accurate than comparing predicted height to data.



Lab 3: Rotational Motion

The first three labs (once three parts to one lab, but broken up to make them doable over two weeks) all use the same apparatus. “Torque and Angular Acceleration” and “Work and Energy in Rotations” can take a decent amount of time so it is usually good to do them on separate weeks. I have found that attempts to speed them up by using stopwatches and multiple trials of the falling mass (rather than lengthy video analysis) did not actually make the labs faster.

Lab 3.1: Torque and Angular Acceleration

Equipment:

- Rotational dynamics apparatus (includes spools on shaft, rotating disk, , extra ring, pulley, string, and mass set)
- Meter stick
- Stopwatch (not mentioned in student manual)
- Video camera

The approximate values for the various items:

Name	Disk	Ring	Shaft	Spools
Mass	1364 g	1431 g	222.1 g	---
Radius	11.4 cm	5.3/6.3 cm	0.67 cm	0.85/1.25/1.85
I (equation)	$\frac{mR^2}{2}$ on its side $\frac{mR^2}{4}$	$\frac{m}{2}(R_1^2 + R_2^2)$, When offset add mx^2	$\frac{mR^2}{2}$ (insignificant)	Very insignificant
I (value)	9.0e-3 kg m^2 4.5e-3 kg m^2 on side	4.9e-3 kg m^2	4.7e-6 kg m^2	----

While masses may be hard to confirm (too massive for scales, etc), students should still measure the radii.

The first rotation labs uses the Pasco rotational dynamics apparatus. As always, make sure it is set up properly, but this lab is otherwise straight-forward.

Equations:
in general:

$$\alpha = \frac{\mathrm{a}}{R}, \;\; \omega = \frac{v}{R} = \frac{2\pi}{t}$$

Where ‘t’ is the period and ‘R’ is the radius.

For the spinning disk

$$\tau = I\alpha = FRsin\theta$$

Where F=T, the tension from the string, and θ is 90 degrees (so $I\alpha = TR$).

For the falling mass

$$m\mathrm{a} = T - mg, \quad T = m\mathrm{a} + mg$$

Thus:

$$I\alpha \rightarrow I\frac{\mathrm{a}}{R} = (m\mathrm{a} + mg)R$$

$$\mathrm{a} = \frac{mgR^2}{I - mR^2} \;\; OR \;\; \alpha = \frac{mgR}{I - mR^2}$$

You do want to make sure that the torque being applied is large relative to the frictional torque on the bearings. Essentially, if the mass is too small, the lab does not work well. Given the speed of the cameras, which you are using to record the acceleration of the falling mass, there is plenty of dynamic range. A mass around 250 g is sufficient and will fall for about 4 seconds.

you should be videotaping the falling mass. The linear values are easier to get from graphs than rotation ones (i.e. fitting $= A\cos(B + Ct + Dt^2)$) . You should get a lot of data points. Because of how slow the mass falls, there will be a lot of uncertainty from point to point. Plotting velocity=x2-x1/t versus time will look terrible (and trying to get acceleration from velocity is even worse). Your best method of finding acceleration is to fit the x vs t plot.

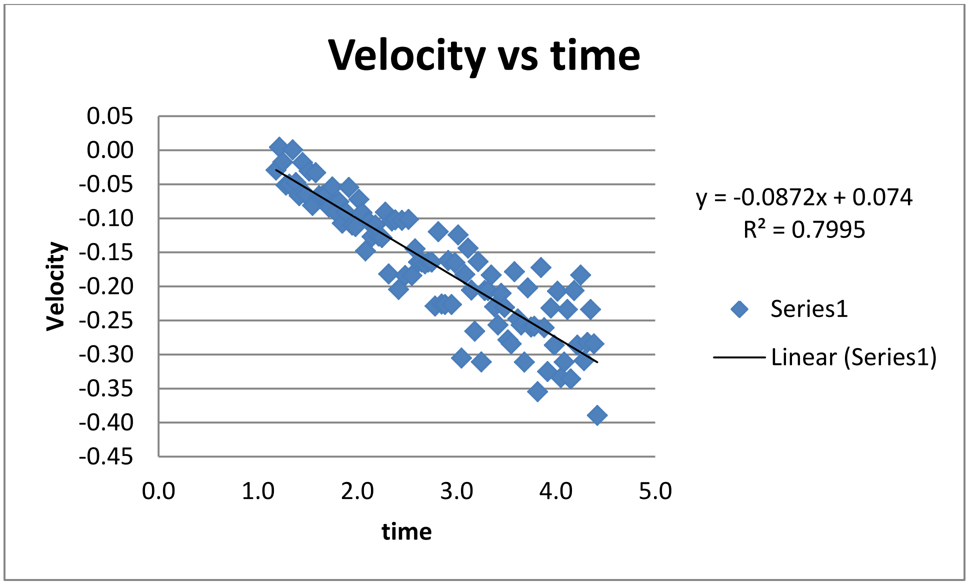
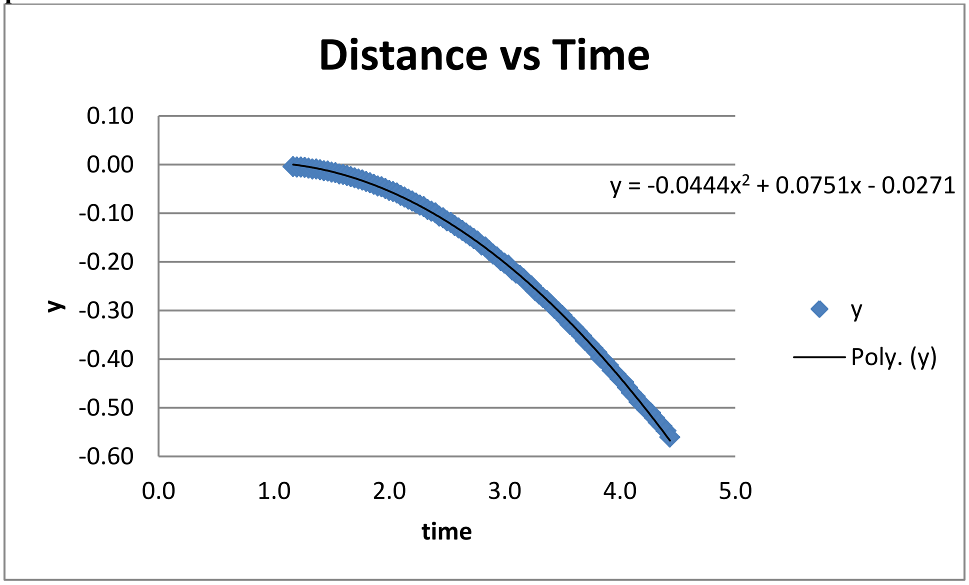


Figure 4: More than a little uncertainty when plotting velocity vs time, position has enough data points to make it a much better choice.



NOTE: the acceleration is twice the coefficient of the second order term! A majority of my students reported half the correct value for the acceleration in their lab books and could not figure out why it didn't match prediction. The acceleration from the above y vs t plot is .0888 m/s^2, NOT 0.444 m/s^2.

I (kg*m^2)	Meas α (rad/s^2)	Pred α (rad/s^2)	meas a (m/s^2)	pred a (m/s^2)	% error
9.03E-03	4.852	4.880	0.0888	0.0893	0.56%
4.52E-03	8.667	9.844	0.1586	0.1802	11.96%
1.40E-02	3.486	3.146	0.0638	0.0576	-10.80%
1.89E-02	2.819	2.326	0.0516	0.0426	-21.22%

For a hanging mass of m=0.2432 kg and spool radius r=0.0183 m (both linear and angular acceleration will have the same percent error, it may make sense to only calculate one of them.

Lab 3.3: Conservation of Angular Momentum

Angular Momentum conservation:

$$I_i \omega_i = I_f \omega_f$$

put the camera so it is directly over the spinning disk. One video should be used to get the before and after angular velocities. Either fit an x vs t graph with a sin function or calculate period to find the angular velocities before and after. The section is by far the simplest.

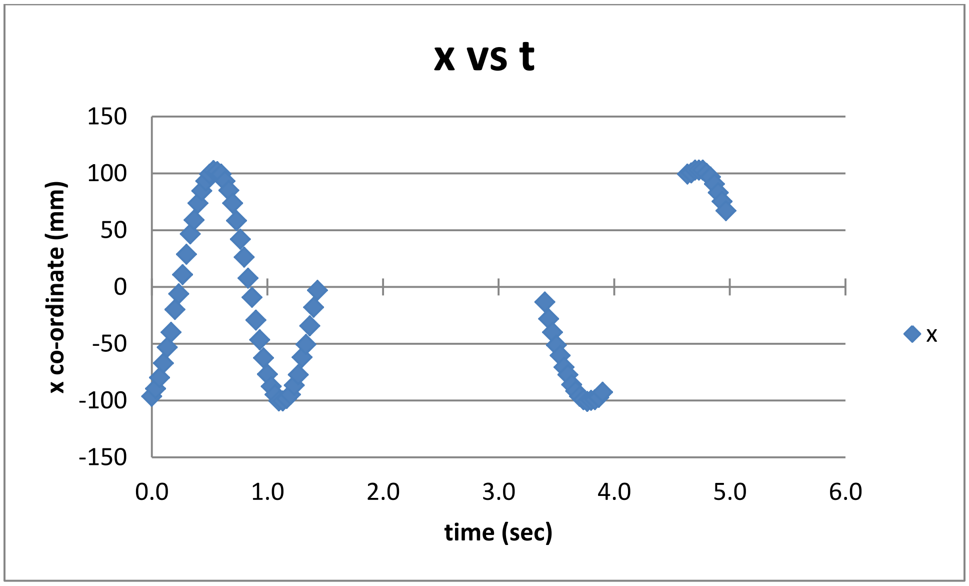


Figure 5: The angular velocities can be found by the max-min time (half a period) or fitting a sin function to each section (not shown). The first section is before the ring is added. The second section is after the ring is added. The gap in the second section is from when the ring blocked the camera's view of the reference point, this is okay as long as there is enough to make a measurement with.

w0 (rad/s)	L0 (kg*m^2/s)	wf (rad/s)	Lf (kg*m^2/s)	% error
5.271	0.0476	3.443	0.0481	-1.0%

I0=9.03E-3 kg*m^2, If=1.40kg*m^2

Lab 3.2 Work and Energy Conservation in Rotations

This lab has the mass falling the same distance, but students changing where the string attaches, thus altering the 'R' variable.

Using energy conservation:

Work=kinetic energy + rotation kinetic energy

$$mgh = \frac{1}{2}mv_f^2 + \frac{1}{2}I\omega_f^2$$

$$\omega = \frac{v}{R}$$

$$v_f = R\sqrt{\frac{2mgh}{I + mR^2}}$$

The idea is that they compare velocities after the same amount of distance is traveled, not simply the final measured velocity of that particular run. It will be impossible to be precise, (ie no two videos will have a frame at the same distance), but certainly close enough to see the intended behavior. When calculating the velocity either fit the x vs t and take the derivative or use the v=dx/t method and take the average value of velocity on a small range centered on the desired one (sample data follows):

trial	Radius	Distance fallen	Predicted velocity	Measured velocity
Spool (large radius)	1.85 cm	23.5 cm	0.205 m/s	0.211 m/s
Spool (small radius)	0.85 cm	23.6 cm	0.094 m/s	0.085 m/s
disk	11.5 cm	22.6 cm	1.079 m/s	1.100 m/s

Lab 3.4: Gyroscope

Equipment:

- Cue ball with embedded magnet
- Aluminum post and sliding weight that can be inserted into the post on the ball
- Air bearing mounted on a base plate at the correct height
- Fish tank air pump make sure each set-up has hoses and fittings, and that the hose is long enough so that pump can be put out of the way
- Video camera
- Ruler
- Stopwatch

Miscellaneous for room: Sharpie markers should be used to make a reference line on the ball.

VERY IMPORTANT: The mass of the weight is 1.5 g, and the mass of the long graphite rods is precisely 1.0 grams (with an uncertainty of about 0.1g). Weighing the mass and rod will likely result in NOT these values, because the uncertainty of the scales isn't that good. **Using any value but these with throw off your final measurements. If students are not getting good agreement, check that they are using these masses.**

Take the air bearings out of the Helmholtz coils to improve access to the experiment.

You should practice to make sure that you know how to get the ball spinning well and how to correct its wobble with a pen or the tip of your finger. Rotational frequencies between 2 and 8 revs/sec. are fine. The precessional frequency is of course much lower. If the air bearing is working properly, the ball should spin for several minutes on its own. If there is a problem, consult Sean Albiston

The intent is for the students to measure the spin frequency of the ball with the camera and the precession with the stopwatch. They must make sure that the frame rate is set to 1/30 second. Yes, they could do it all with the camera, but that will be tedious.

The math can be found by searching for information for precession but the final results are:

$$\Omega = \frac{\tau}{I\omega \sin\theta}$$

The students should break down the prediction into:

$$\Omega = \frac{mlg}{\left(\frac{2}{5}\right) M_{ball} R_{ball}^2 \omega}$$

(a factor of sinθ cancels from top and bottom of the fraction)

‘mlg’ consists of both the weight AND the rod. Depending on the length of the rod, it can be the dominating factor.

Item	Mass (g)	Distance CM to CM (cm)	Torque Contribution (N*m)
Weight	1.5	4-14	4.3e-4 to 1.5e-3
Rod	1.0	7	6.9e-4

To account for this the final equation should look like:

$$\Omega = \frac{m_{weight} l_{weight} g + m_{rod} l_{cm} g}{\left(\frac{2}{5}\right) M_{ball} R_{ball}^2 \omega}$$

Since you cannot ‘control’ values for ω, you should plot $\Omega\omega \; vs. \frac{\tau}{I}$.

IMPORTANT NOTE: This equation reportedly works best when ω >> Ω. Students should try to get the ball spinning as fast as they can accurately measure with the camera. This will have the benefit of slowing down the precession making it easier to measure, and seems to keep wobbling (nutation) to a minimum.

IMPORTANT NOTE: While according to the equation the angle the rod is at (θ) should not have any effect, it apparently still does. Have students keep an approximately constant θ between trials or they will find data frustratingly unrepeatable. Smaller θ are likely the safest way to go.

While this lab is easy to do and the results are qualitatively easy to see, it can suffers quantitatively. Two seemingly identical trials can have noticeable variation. I was unable to find a perfect way to do things, but you can at least get close.

Problem shooting:

1. Make sure you use a value of 1.5 g for the weight and 1.0 g for the rod, regardless of what a scale tells you.
2. Spin the balls fast.
3. Make sure the students have included all relevant torques
4. Keep θ constant between trials.
5. Make sure the wobbling is minimized.

6. Do not add more weight. Increased weight will reduce stability faster than it will reduce other errors.

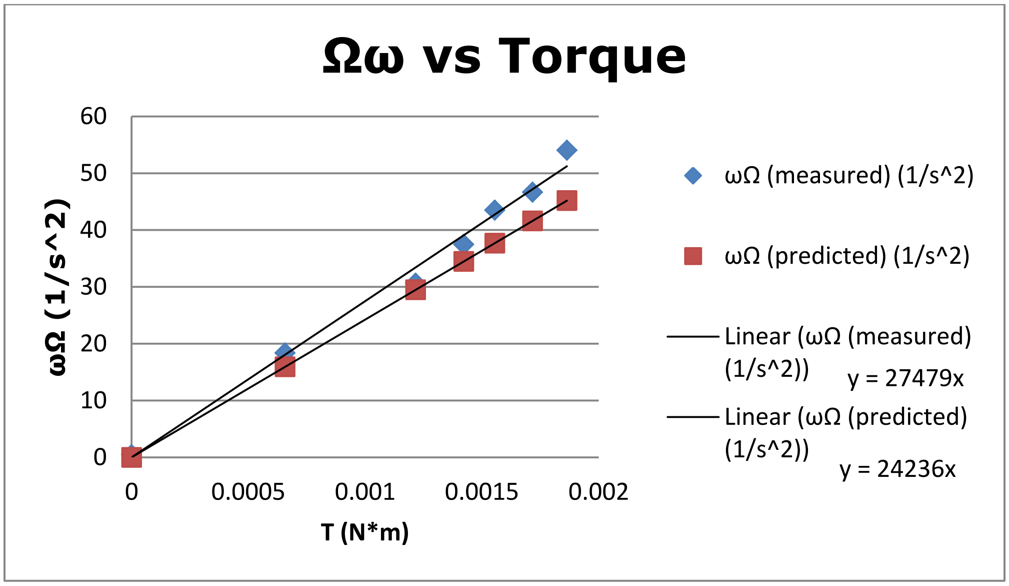


Figure 6: The best trick for getting prediction and measurements to agree is setting the mass of the weight to 1.5 g and the rod to 1.0 g.

Nutation of a top:

The nutation section is for students who finish early to further explore the properties of the gyroscope. The complete formulas for nutation are quite complex, so mostly qualitative answers can be given. Things students might note:

1. Faster spin decreases nutation.
2. very small or very large tilts seem to decrease nutation, the worst nutation shows up at medium angles.
3. Nutation increases with time, especially when slow spin and medium tilt.
4. The Earth nutates (feels torque from the sun and the moon).

Lab 4: Gravitation Simulation

Appendix A contains samples of the code useful for each section.

Lab 4.1: Gravity Simulation: Introduction to VPython

Purpose:

The purpose of this exercise is to create a simulation of two objects orbiting each other due to gravity. Use the simulation to verify Kepler's laws.

Materials:

You will need a computer with *Python* and *V-Python* installed. These and documentation can be found at www.python.org and http://vpython.org, respectively.

Obviously, you must be completely familiar with *Python and VPython* before the students do this laboratory. You have many weeks to prepare, and so there are no excuses. The lab itself is a simple exercise in numerical integration.

!!!Versions of code that should replicate each step can be found in Appendix A!!!

Basics – Programming in Python:

This exercise requires only some very basic programming skills and then learning a couple of very easy commands. Python is a fairly popular object oriented language, but most likely you have exposure to C/C++, FORTRAN or Java. The syntax of Python is very similar to C/C++ and FORTRAN, but if you know any programming language, this should find this straight-forward.

We can't go through a complete tutorial of the Python language. If you are interested, you can pick up any "Learn C [or C++ or FORTRAN] in 21 Days" text, or stop by the book store and pick up one of the texts in the CSci section. Any of these choices are great as far as learning how the syntax works together. Take a look at the tutorials on www.python.org, or check out the documentation on special functions at http://vpython.org/webdoc/visual/index.html. There are also several free tutorials and introductory books available at www.freeprogrammingresources.com. Still, this tutorial should be enough to get by with, and we give you all of the code to make a proper program.

Starting a Program

Begin each program with a header. This header should load any necessary library files to the compiler so that your program can be correctly interpreted. The header that we will use is to load the V-Python libraries, and is given below.

```
from visual import *
from visual.graph import *
```

There are ways to make comments and leave notes for yourself or other programmers. Begin the comment with the "#" symbol. These are very useful and you will notice that we make prolific use of them in the examples to follow. An example of a comment is given on the next page.

```
# This is a commented line.  Only text on this line will be ignored by the
# compiler.
However, text on this line will produce an error because it is not commented.
```

You will need objects to work with in the program, and V-Python is useful for this. You can declare objects such as spheres that have attributes such as position (pos), velocity (vel), radius, color and mass. For example, we can make a spherical object "satellite1".

```
        Satellite1 = sphere(pos(-5,0,0),radius=0.1,color=color.blue,mass=1)
```

We will discuss how to manipulate these objects a little later on.

Next is the body of the program. This is where you give commands for the program to do. The compiler reads everything from top to bottom and left to right, so the order that you place certain functions will be important. We will discuss how to visualize or output data from our program, but first let's just talk about making it do some calculations.

Doing Math in Python

Math in Python looks just like it may on paper. You can add, subtract, multiply and divide as usual, and Python follows the usual order of operations.

There are a couple of things that you should be aware of about performing calculations:

A single equals sign (=) is the assignment operator. The variable to the left of it is the variable being assigned the argument on the right. Consider the quick following example.

```
                    radius = circumference / (2*3.14)
```

The variable "radius" is being assigned the value from the calculation on the right.

Two equal signs (==) is an evaluation of the two adjacent arguments. For example, the statement "radius==circumference" evaluates whether the two values are equal or not. If the equality is true, then the function will return the value of "1", and "0" if untrue. Similarly you can evaluate inequalities (<,>,<=,>=). Usually you will use these in loop functions or gates.

Python reads everything from left to right. Don't worry about your parentheses – again order of operation still holds true here, and it will do the calculations correctly. However, you cannot assign the value 12 to the variable "x" written as the statement below.

```
12 = x
```

This simply won't work. The variable being assigned must be on the right, and the value on the right. A correct statement should be like that below.

```
x = 12
```

Another set of common operators that you may use are the power, square root and exponential operators. Trigonometry operators such as the sine, cosine and tangent functions are the same as you would write on paper, and their inverse functions are simply noted by putting an "a" in front of the original operator. Examples of these are shown in the table on the next page.

Function	Code	Output
Exponential	exp(x)	e^x
Logarithm	log(x,base)	$\mathrm{Log}_{base}(x)$
Power	pow(x,y)	x^y
Square root	sqrt(x)	$\sqrt{x}$
Sine	sin(x)	Sin(x)
Arcsine	asin(x)	$\mathrm{Sin}^{-1}(x)$

V-Python has made a very useful set of objects, operators and functions that make a lot of the tedious programming routines easier, primarily for vectors.

To create a vector you need a name for the vectors (say v1 and v2), and then some components (x,y,z). The statement below shows how you can initialize or assign two vectors.

```
v1 = vector(1,2,3)
v2 = vector(4,5,6)
```

These vectors obey normal scalar addition, subtraction and scalar multiplication with the usual symbols (+,-, and *). Below are examples of the cross- and dot- product operations.

```
cross(v1,v2)
dot(v1,v2)
```

A couple of other useful operations are ones that give you the magnitude (length) of the vector (v1 in the example below), and another that gives the angle between two vectors (v1 and v2), shown respectively.

```
mag(v1)
v1.diff_angle(v2)
```

That's it. This is really all the *math* that you will need to complete the program. Now we will discuss the theory we need to implement, and then how to go about programming it.

Theory:

The programs that you and your students will create will demonstrate the motions of objects in *elliptical* orbit over time.

We will consider two objects with masses *M* and *m*. Suppose that the mass M is fixed in space, and the second mass, m, is initially a distance *R* away from the first. In order for the smaller mass to not fall into the larger one, we must give it some initial velocity v_o. For simplicity, we'll make this velocity perpendicular to the line that passes through the two masses.

You may wish to have your students start with circular orbits – they should be quite accustomed to these by now, and it will make the programming process go a lot easier. The condition for the circular orbit is that the centripetal force experienced by the orbiting object is equal and opposite to the gravitational force.

$$-\frac{GMm}{R^2} = m\frac{v_o^2}{R},$$

where *G* is the gravitational constant. Solving for v_o, and ignoring the minus sign, we see that

$$v_o = \sqrt{\frac{GM}{R}}.$$

This situation is fairly trivial for students at this level, and we don't want them to crutch on this too much. We are trying to simulate actual *elliptical* orbits so that they can confirm Kepler's laws for in a more general and sophisticated way.

Of course we already know that to control our virtual massive objects we will have to use forces and kinematics in our program, but to make elliptical orbits it's best to use energy to describe what is really going on.

The potential energy, *U*, of the smaller object, *m*, at a distance *R* from the larger object is given by

$$U = -\frac{GMm}{R}.$$

The negative sign is a consequence of the gravitational force being attractive and that we chose the potential energy to go to zero as mass *m* moves infinitely far away. The total energy of the system, *E*, is the potential energy plus the kinetic energy:

$$E = \tfrac{1}{2} m v^2 - \frac{GMm}{R}. \qquad \text{(a)}$$

One can now see that for circular orbits, the total energy is

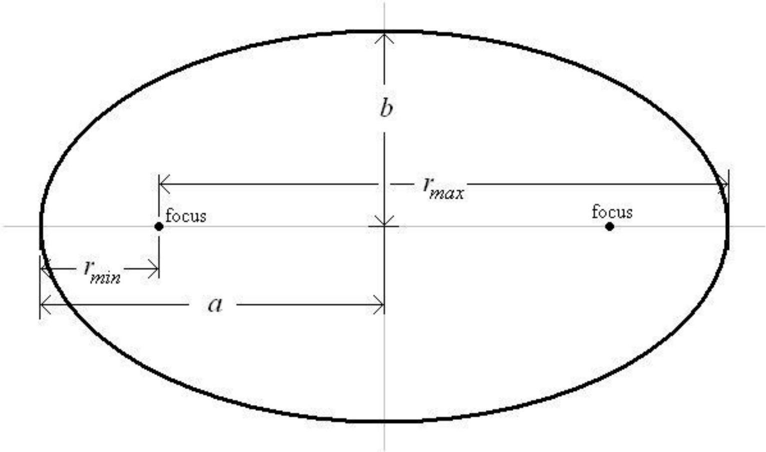
$$E_{circular} = -\frac{GMm}{2R}.$$

Again, we are more interested in *elliptical* orbits. The condition for making an elliptical orbit is that the total energy must be greater than that for a circular orbit, but less than zero:

$$-\frac{GMm}{2R} < E < 0.$$

If we substitute equation (a) for *E*, and solve for the velocity squared, we find that

$$\frac{GMm}{R} < v^2 < 2\frac{GMm}{R}. \qquad \text{(b)}$$



This is all fine and dandy, but this is not the only condition – we still need to know about the initial conditions and parameters of our elliptical orbits. Consider the diagram to the right. The ellipse has two axes – the major axis with a length 2*a*, and the minor axis with a length 2*b*. There are also two points labeled as the foci. Two distances that will be important to us are the maximum distance or apogee ($r_{\max}$) and minimum distance or perigee ($r_{\min}$) from one focus.

A characteristic value of an ellipse is its eccentricity, *e*. This is given by

$$e = \frac{r_{\max} - r_{\min}}{2a}.$$

With a little bit of algebra, and the fact that $r_{\max} + r_{\min} = 2a$, then

$$e = 1 - \frac{r_{\min}}{a}. \qquad \text{(c)}$$

The eccentricity can be used to describe circles, ellipses and parabolas; however, for an ellipse, the eccentricity always must have a value between zero and one:

$$0 < e < 1. \qquad \text{(d)}$$

Finally we are ready to make our elliptical orbit. If we begin our simulation with the sun at a focus of the ellipse and the smaller mass at the perigee of the orbit, then using equations (b) and (c) we find that the velocity, v, of the orbiting object must meet the condition that

$$\frac{GM}{a(1-e)} < v^2 < 2\frac{GM}{a(1-e)}. \tag{e}$$

Therefore, to make the elliptical orbit, you will only need to *specify* a few things before you begin running your program:

- the masses of your two objects (*m* and *M*),
- the initial distance between the two objects (r_{min}),
- and the eccentricity of the orbit (*e*).

You can use equation (c) to determine the value of *a*, and be sure that you still satisfy the condition in equation (d).

Implementation:

Now we can actually start writing our program.

Start out with the header discussed earlier. Then you will need vectors to describe your object's positions, velocities, and accelerations. You may even want to add in angular momentum for each of the objects, since you will need them in the near future. A vector for the center of mass may be helpful, too. You will have to determine many of these values and vectors beforehand using equation (e) and the list of specifications on the previous page.

You will need scalar variables such as those that describe distances, mass, time or any constants. The example below was made to give you an idea of how to go abut doing this – it is not complete for the program.

```
# Begin with the header
from visual import *
from visual.graph import *

# Now declare the two objects for the satellite and sun
satellite = sphere(pos(-100,0,0), radius=1, color=color.blue, mass=1)
sun = sphere(pos(0,0,0), radius=100, color=color.red, mass = 10000)

# Give the satellite attributes like velocity and acceleration and distance from
# the sun
satellite.vel = vector(0,0,0)
satellite.acc = vector(0,0,0)
distance = sun.pos-satellite.pos

# Initialize scalar variables and constants for the time steps (dt), number of
# steps (maxsteps), the current step, (tstep), total time that you can count up
# (tTime), and G.
dt = 0.01
```

```
maxSteps = 6000
tTime = 0
tStep = 1
G = 6.67

# Make any blank lists that you need by making an array – specify this with "[]"
posArray = []
```

It's best to declare and initialize all of your variables at the beginning of your program. There will be some variables that you will have to specify in order for the simulation to work, such as the initial separation of the objects, velocities, and masses.

Next you will have to calculate the acceleration of the satellite, and its displacement in the small time, dt, as in the example above. These calculations can actually be quite simple. Note that you don't need to keep writing "vector" each time. Once a vector is initialized, only need to refer to it by name. If you want a specific attribute of an object, add a period and name of the attribute after the object's name. For instance, to recall the position of the satellite, simply type "satellite.pos". See the example syntax below.

```
# Calculate the acceleration of the satellite
satellite.acc = (G*sun.mass)*distance/pow(mag(distance),3)

# Now use kinematics to determine new position of the satellite after time dt
satellite.pos = satellite.pos + satellite.vel*dt + 0.5*satellite.acc*dt*dt
satellite.vel = satellite.vel + satellite.acc*dt
distance = satellite.pos-sun.pos
tStep = tStep + 1
tTime = tTime + dt
```

It is important to remember that for every time that you make a calculation, you will have to update all of your variables that change with time.

To do these steps over and over, you will need the help of loops. There are a couple of choices of the kind of loop you may use, but to keep things simple we will use the "for" loop. This command loops through everything indented below it for as many times that you specify. In the example syntax below, we repeat the commands described above as long as the variable "tStep" is between the numbers 1 and 6000 (specified by "maxSteps").

```
for tStep in range(1,maxSteps):
        # Calculate the acceleration of the satellite
        satellite.acc = (G*sun.mass)*distance/pow(mag(distance),3)

        # Now use kinematics to determine new position of the satellite after
        # time "dt"
        satellite.pos = satellite.pos + satellite.vel*dt + 0.5*satellite.acc*dt*dt
```

```
    satellite.vel = satellite.vel + satellite.acc*dt
    distance = satellite.pos-sun.pos
    tStep = tStep + 1
    tTime = tTime + dt

    #You can add data to an array, such as the x-position
    posArray.append(satellite.pos.x)

# The next comman or any below it will not be repeated in the loop because it is
# not indented.  Only the commands above will be repeated.

print posArray
print “completed”
```

The last command here will print the values in the array to another screen. The arrays are useful for filling with data you will want later on, and even groups of numbers can be stored as one unit if you put then inside another set of parentheses, such as (12,552), for instance. Other things can be printed in the window as well, such as text that is enclosed in quotations, such as the word "completed" above.

Visualization:

The "print" command is one way to generate an output from the program, but it is a bit limited. There are a number of things that will help see what is going on.

For one, you can make a trailing curve that will draw out the path of an object in the 3-D output screen. The attribute for the object is called "trail", and you simply need to give it a color and update it as you do other variables.

```
# This is how you will initialize the trail array
satellite.trail = curve(color=satellite.color)


# To see the trail, the trail variable must contain all of the previous positions of
# the object.  Just tack on the new position using the “append” command (the
# trail variable stores this in its own “pos” attribute)
satellite.trail.append(pos = satellite.pos)
```

Its nice to see the orbs go around in the animation, but we are really interested in the data. To make a graph of the position of your satellite, you need to initiate a plot, preferably right after you initialize your variables near the beginning of the program. The following syntax will do the trick.

```
# Setup graph (axes, size, labels, ranges and colors)
trajplot = gdisplay(x=0, y=0, width=800, height=400,
                    title=”y vs x”, xtitle=”x position”,
                    ytitle=”yposition”, xmin=-5.5,
                    xmax=5, ymin=0, ymax=25,
                    foreground=color.black, background=color.white)


# At the end of the program, when you are ready to plot your points, such as
# those in the array “posArray”, type in the following command.
xyplot = gdots(pos=posArray, color=color.blue)
```

For a plot with two axes, the data in the array must be grouped by parentheses. You can easily group data, such as the x- and y- positions of the satellite, in the array as shown in the following.

```
posArray.append((satellite.pos.x,satellite.pos.y)
```

Lab 4.2: Verifying Kepler’s Laws

Data and Analysis:

From your simulations you will need to verify Kepler’s laws of orbits, areas and periods.

The Law of Orbits

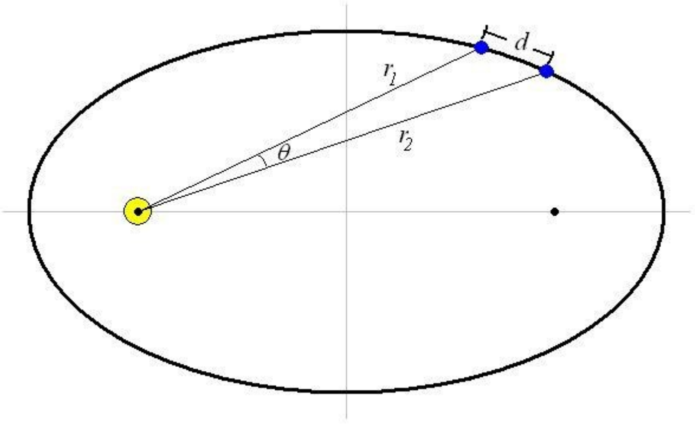
Have the students mess around with the initial velocities and positions in their programs. Use equation (b) and see what happens when you don’t satisfy the inequality or the condition for circular motion. Have them note that the only stable orbits draw out an ellipse or a circle. This is a simple qualitative establishment. The trick is finding quantitative data to verify their predictions.

One of the easiest ways to do this is to find the satellite’s distance from each focus of their ellipse (if that is indeed what the orbit is). The sum of these two distances should be equal 2*a*. You may even have them determine the values of *a* and *b* in the equation for an ellipse:

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

The Law of Areas

This section is tricky because the students need to calculate the area for each displacement, and some of them may be fixated with the arched ends of each section of the orbit as shown in the figure to the



right. We can simplify the problem by choosing small displacements (which means small measures in time).

To determine the area of each small triangle swept out in a small time, *dt*, we will have to assume that the distance the satellite has moved is in a straight line, and not an arc. The displacement can be found from your usual kinematic equations, but the easiest way to find the area is to compute the cross product of the two vectors, r_1 and r_2. If you wish, you can find the angle, *θ*, between the two distances as shown above. The area, *A*, of the small triangle is

$$A = \tfrac{1}{2}\left\|\overrightarrow{r_1} \times \overrightarrow{r_2}\right\| = \tfrac{1}{2} \cdot r_1 \cdot r_2 \cdot \sin(\theta)\,.$$

In theory, if you look at the conservation of angular momentum you should be able to show that the area swept out, *dA*, in a small period of time, *dt*, is equal to the angular momentum of the satellite, *L*, divided by twice its mass, *m*:

$$dA = \frac{L}{2m}dt\,.$$

This is the value you should try to confirm with your program.

The Law of Periods

This section may take some time because it involves running several simulations. Each simulation should consider a new orbit – different mean distances from the sun and/or different speeds at the perihelion. Have the program print out the time to complete one revolution around the sun for each of the simulations. You should also record the length of the semimajor axis, *a*, of each ellipse that you draw out. Make a plot of the period squared against *a* cubed. This should be a linear relationship.

	Position	Velocity	Period	Semi-major axis	Period^2/Semi-major axis^3
Trial 1	(-5,0,0)	(0,5,0)	1.1165	2.758636129	0.05937918
Trial 2	(-9,0,0)	(0,5,0)	3.0635	5.412510858	0.059188785
Trial 3	(-5,0,0)	(0,7,0)	1.3045	3.062175183	0.05926497

Angular Momentum

To confirm the conservation of angular momentum, have the students store the satellite's angular momentum over each iteration in their programs along with the associated times. After they run the simulation, have them plot the angular momentum as a function of time. It should be a reasonably flat line. Of course the law of areas *is* conservation of angular momentum. If students put angular momentum in as an arrow note it may point directly at them. They can rotate the image by right clicking and moving around.

Precession of the perihelion

Students can output the position of the perihelion and run the orbit multiple time to see the small change for the r^2 case:

```
perihelion at <1.12536, -0.0627435, 0> .
perihelion at <1.12548, 0.0452614, 0> .
perihelion at <1.1263, -0.00238513, 0> .
perihelion at <1.12607, -0.0500257, 0> .
perihelion at <1.12487, 0.0579715, 0> .
perihelion at <1.12628, 0.0103387, 0> .
```

For the different powers (1, 2.5, 3) the can change their definition of gravity and run using the same output. However, visually they will see the differences almost immediately.

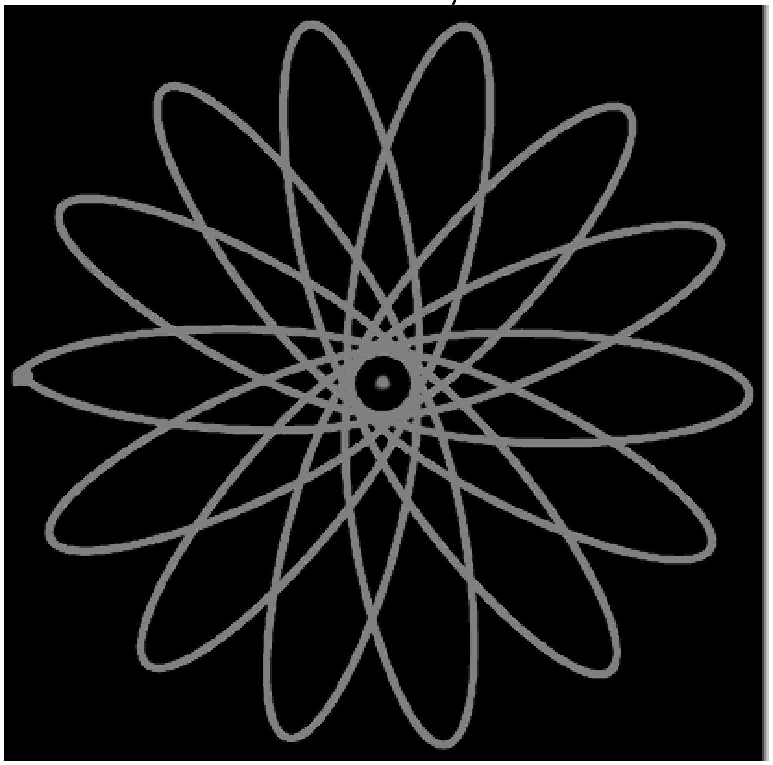


Figure 7: using 1/r, the precession will repeat once the 'pattern' is complete.

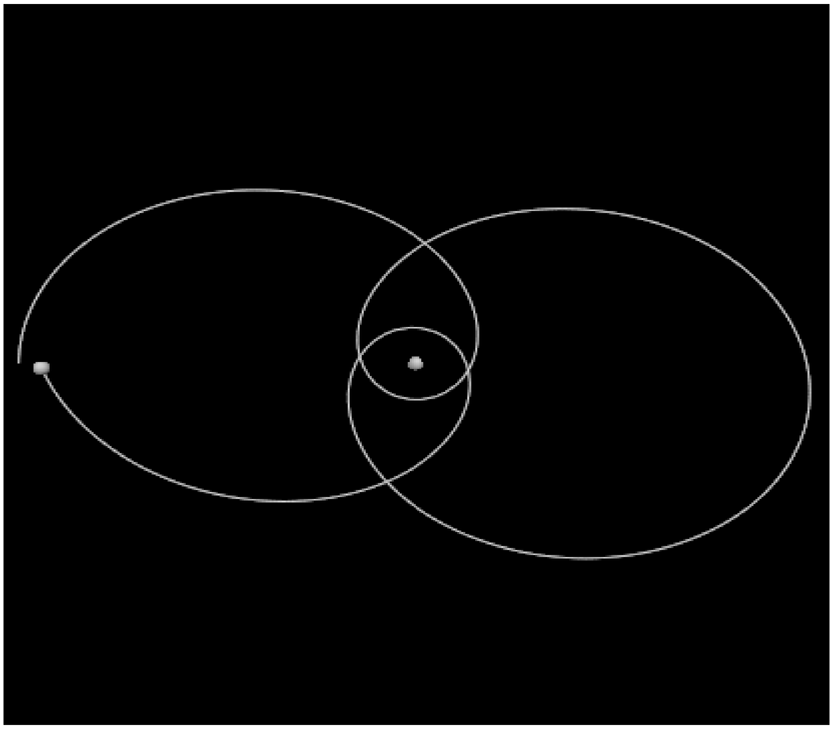


Figure 8: using 1/r^2.5, as the force gets weaker, orbits are more eccentric

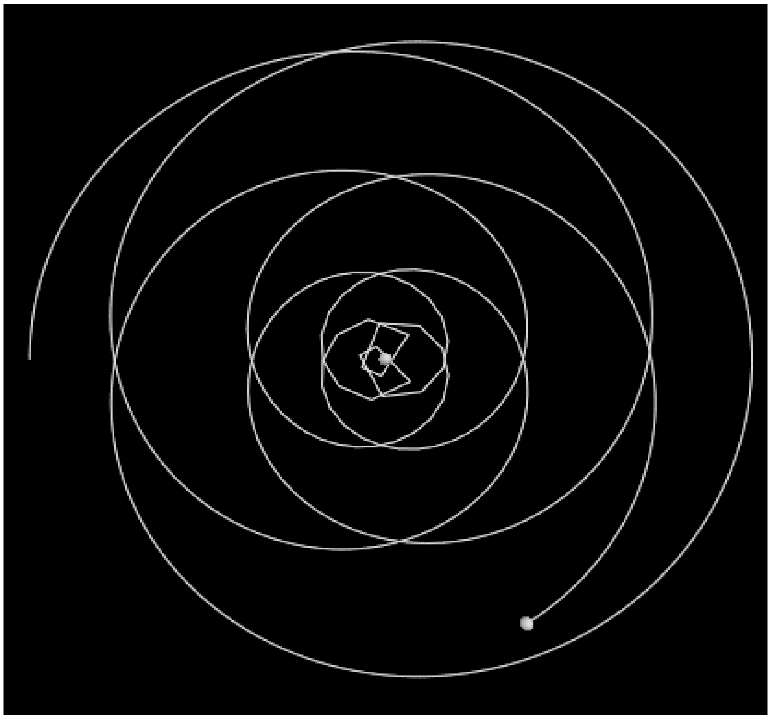


Figure 9: when force varies as 1/r^3 student will be hard pressed to get a stable orbit. Most either escape orbit, or fall into the sun. If they do get it stable it will spiral closer, then gain speed and get slung back out.

Binary stars

Of course they can just change coordinates and replace the mass with reduced mass and claim they are done! The intent is for them to program it without changing coordinates and then see that both orbit the center of mass.

Lab 4.3: Advance Simulation: The Tides

By far the most challenging portion, you should consider letting this section be optional. Often one or two groups will have a student that is particularly good at programing and will finish much earlier than the rest. This is a good task for those groups.

Have students start with just an earth/moon two body problem. Get the moon going around the earth; looking up moon's distance and velocity at certain points in its orbit should help. Then need to define a point that rotates around the earth at its radius (to mimic the point being on the earth's rotating surface). You can tell if the Moon's gravity and the earth's gravity are co-linear if their dot product is maximal (or when their cross product is 0), and they are perpendicular if their dot product is zero.

Lab 5: Simple Harmonic Motion

Lab 5.1: Damped Simple Harmonic Motion

Purpose:

This lab explores damped simple harmonic motion. The students can investigate the dependence of the frequency on mass, measure damped oscillations, and measure the relationship between the frequency and the damping constant. The damping constant is varied by changing the number of magnets on the bottom of the cart.

Materials:

The following materials may be used for this lab:

- 1 Pasco® aluminum track (2 m in length),
- a Pasco® cart,
- 1 to 4 small NdFeB magnets,
- a power supply (up to 12V),
- 2 springs,
- a Vernier Sonic Ranger® motion detector and a piece of stiff cardboard to mount on the cart (to reflect sound)
- a Vernier SensorDAQ® or LabPro® interface,
- string,
- tape, and
- a computer with Logger Pro®.

Setup:

Set up the cart between the two end stops, with springs attached to both sides of the cart. **Check the motion sensor at each table before the students arrive. As always, if there are problems check the data collection rate. Also, remember that the sound needs to be reflected off of a card mounted on the cart. Sometimes the sensor picks up a reflection from something on the next table.**

One of the problems with this lab is that students do not record enough cycles of motion. Try to obtain data over two time constants. They need to be able to make a log plot of amplitude vs. time.

Note that the magnets provide a nearly perfect linear damping mechanism due to eddy currents.

Procedure:

1. First, verify that the resonance period depends on mass in the correct way and that the damping is small. You should be able to observe many oscillations. Although not mentioned in the instructions, the students should vary the mass. Remember to check for bad carts.

2. Mount **one** of the magnets on the bottom of a cart by placing it over a metal screw. Pull the cart back some distance (about 30 to 40 cm), and begin collecting data with motion detector and Logger Pro®. Release the cart and observe the free decay of oscillations of the cart. **You will need to tape a piece of cardboard or a playing card onto the cart for the motion detector to work properly.**

Save your data to an Excel file, and determine the frequency of the oscillations. **You should shift the x-values so that it oscillates around zero!**

Determine the value of *b*, the characteristic value of the damping constant by making a log plot of the amplitude as a function of time. If your data is oscillating around zero you can use the absolute values of the positive and negative peaks for maximum number of points. The slope will be (-*b*/ 2*m*), where *m* is the mass of the cart. In principle, you can also determine the shift in the resonant frequency

$$\omega = \sqrt{\frac{k}{m} - \left(\frac{b}{2m}\right)^2} = \sqrt{\omega_0^2 - \left(\frac{b}{2m}\right)^2},$$

where ω_o is the natural frequency of the spring-cart system. In practice, shifts in the resonant frequency due to damping will be small. Use the values you get to create a function (from equation 5-9) that matches your measurements. You can also directly measure ω (and solve for b) by measuring the period (*T*) using:

$$\omega = \frac{2\pi}{T}, \qquad b = 2m\sqrt{\omega_0^2 - \omega^2}$$

Have students add more magnets and predict what the new frequency should be. You should also encourage them to vary the mass of the cart for a fixed number of magnets.

Note: *b, ω,* and *T* are constant over the course of one plot, only Amplitude varies with time.

Data and analysis:

With the data students have already, they can make a plot of the frequency against the number of magnets on the cart, or against the mass of the cart for a fixed number of magnets. They should also plot their predictions alongside of the recorded values.

They should determine the characteristic damping time for each of the trials **This may be their first experience with a logarithmic plot!** They can then plot the damping time as a function of the number of magnets.

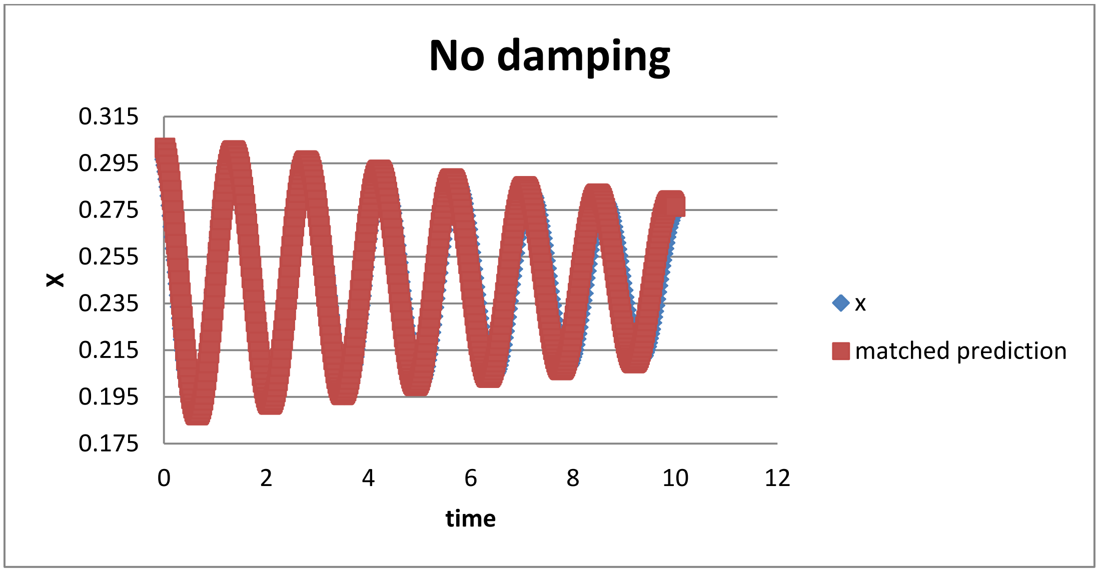


Figure 10: The results with no damping and with the matched precition using only the b value as input (angular velocity is calculated from b). b=0.084, ω=ω0=4.4, T=1.4 sec.

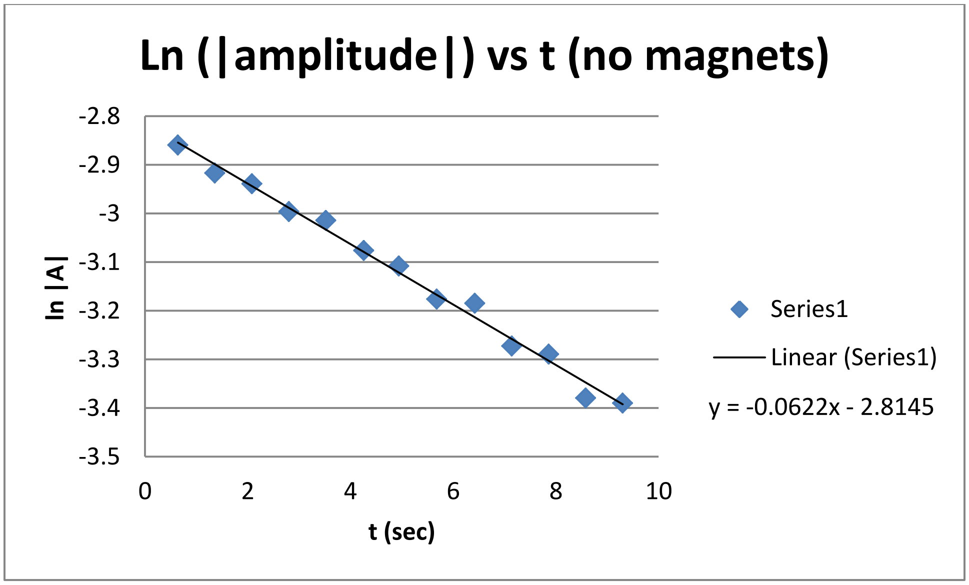


Figure 11: plot of the natural log of |amplitude| vs time for no magnets. The slope of the line is -b/2m. Here b=.084, ω=ω0=4.4, T=1.4 sec. Using the absolute value allows us to use minima in the plot for extra data points.

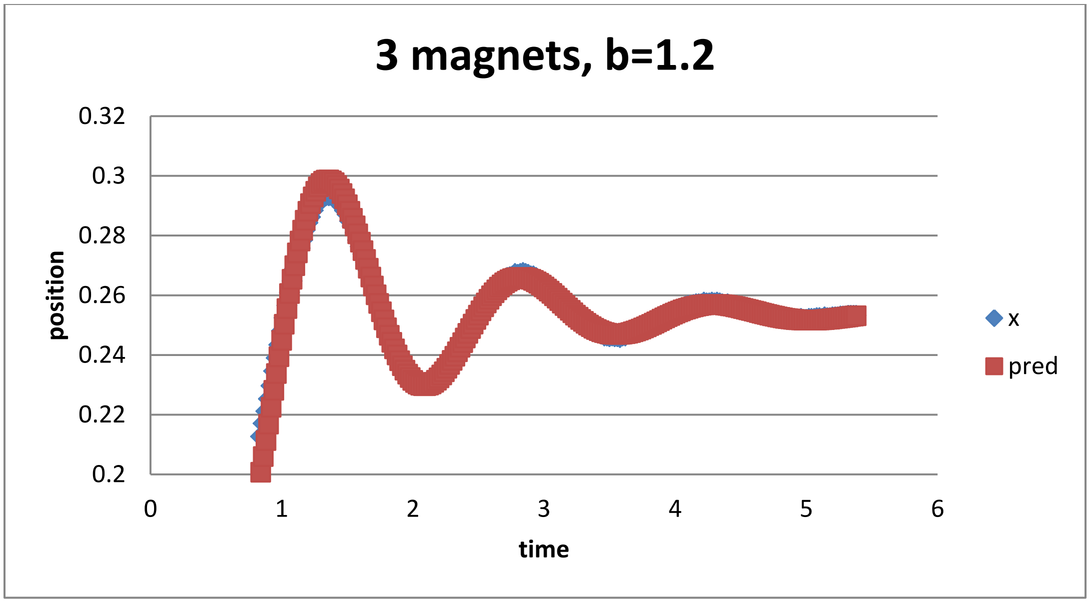
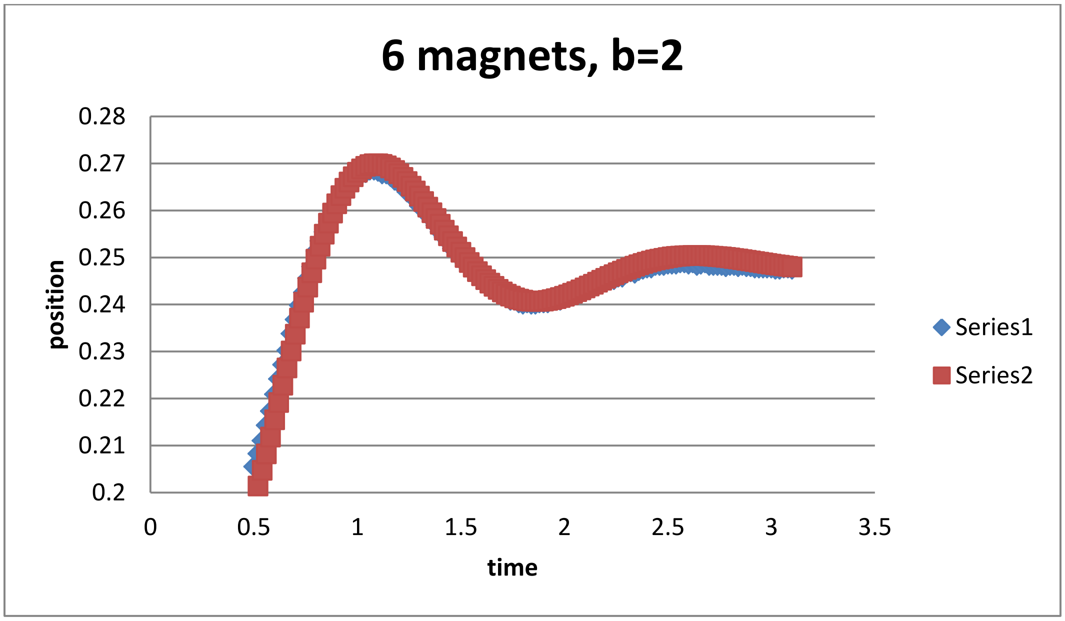


Figure 12; dampening quickly reduces the number of oscillations. Magnets seem to increase b by about 0.2-0.4.



Notes and Comments:

This lab could be useful for practicing error analysis. Some of the students who are more on top of things should be able to determine the standard deviations in their data, and possibly come up with a decent value of *b* and an uncertainty.

Lab 5.2: Driven Simple Harmonic Motion

At a minimum, students should plot the resonance curve as a function of frequency for at least two values of damping. With a little bit of thought, they can also measure the phase.

Note that the instructions for measuring the phase here are different than those in the student lab manual. In fact, the scheme suggested in the student manual is not very practical.

It is best to start with some damping and a reasonable mass on the cart. This keeps the resonant amplitude from getting too large and the damping time from being too long.

Materials:

The following materials may be used for this lab:

- 1 Pasco® aluminum track (2 m in length),
- a Pasco® cart,
- about 4 Pasco® 250 g bricks for the cart,
- a Pasco® mechanical oscillator/driver & mount
- a power supply (up to 12V),
- 2 springs,
- a few NdFeB magnets,
- a Vernier photogate,
- a Vernier Sonic Ranger® motion detector,
- a Vernier SensorDAQ® or LabPro® interface,
- string,
- a ruler,
- tape, and
- a computer with Logger Pro®.

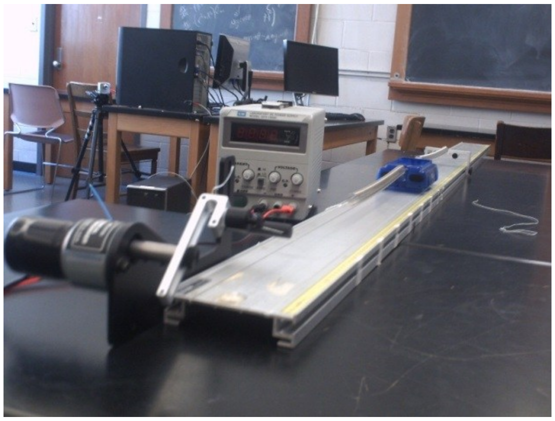
Setup:

Begin by placing two end-stops on the track about 150 to 180 centimeters apart. Place a cart between the end-stops on the track, and attach each end to an end-stop with the springs. You should add a stiff piece of cardboard onto the cart to reflect the ultrasound back into the motion detector.

Without any dampening there are secondary resonances that occur and cause the amplitude to cycle. **Add one or two magnets to the bottom of the cart.**

The students should the dampening and the natural frequency using the same methods from the previous lab. They can add mass to bring the frequency down into a convenient range. Adding mass also seems to increase the amplitude, so it is probably a good idea to start with a few weight on the cart.

We have traditionally used motors to drive the system (set up as shown below). It is easy to adjust the amplitude at low frequencies. Do not use the linear speaker-like driver, it does not have enough force to really move the cart at all.



Once you are done with the procedure above, remove one of the end-stops and replace it with the mechanical oscillator/driver. The motor set should be placed at the end of the track so that it can freely move. Thread a piece of string through the hole in a guide cylinder on the oscillator's mount and tie it to the arm of a linear DC motor. Tie the other end of the string to the spring connected to the cart. Be sure that the spring is stretched to approximately the same length as it was with the end-stop. The total setup should look like that in the figure above.

The oscillators should be driven with our standard 18V DC power supplies, but the **motors are only designed or 12V DC**! Make sure students do not exceed this voltage. The frequency increases with increased voltage, and the amplitude is set by the (adjustable) arm.

The mechanical drivers act as the *forcing* mechanism for our system. The strength of this force is directly proportional to the length of the arm on the motor. Usually an arm length of 2 cm works out well, small amplitudes work best.

Place the Sonic Ranger at the opposite end of the track and plug it into the interface. To measure the phase, you should place a photogate behind the motor. Position this so that the arm of the motor passes through the beam of the photogate. You may have to play around with getting the cords to the interface.

Procedure:

The lab consists of mapping the resonance curve. Have students change the frequency of the driving motor. Allow about 5 seconds (i.e. a few time constants) for the cart to settle at each frequency. Obviously the most interesting data for this lab will be near the resonant frequency.

.

Repeat this procedure for several frequencies. Be sure to cover the three primary cases: $f<<f_o$, $f=f_o$, and $f>>f_o$. It works best to begin at lower frequencies and steadily increase. You should be taking more data points when closer to the resonant frequency.

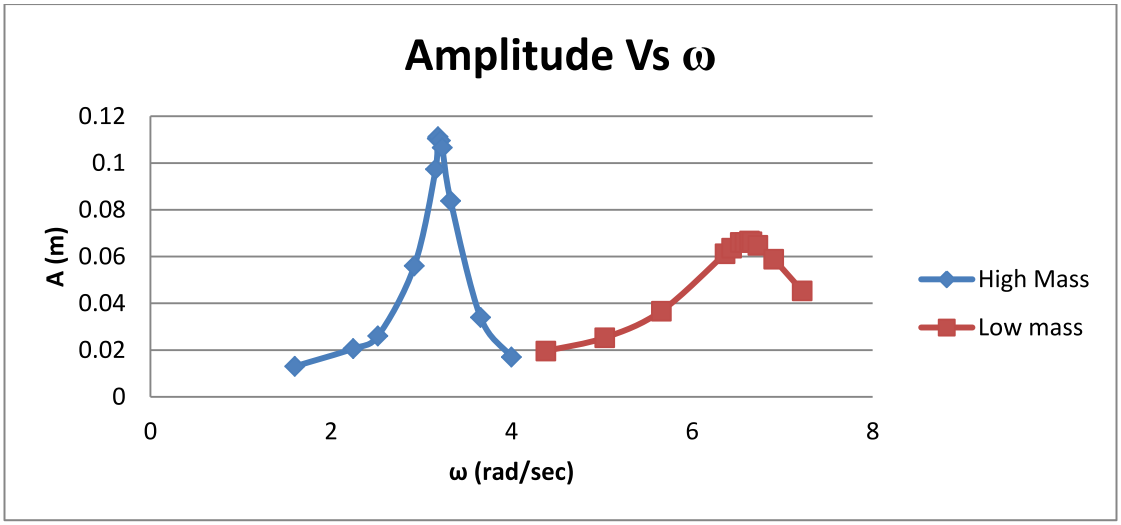


Figure 13: Comparison of the resonance peaks of two trials: low mass, low dampening (b=0.44, m=0.273 kg, predicted ω0=6.93 rad/sec, measured peak 6.64 rad/sec) and high mass, high damping (b=4.04, m=1.273, predicted ω0=3.21 rad/sec, measured peak 3.18 rad/sec).

This is one of the rare labs where phase relation between the drive/oscillator at resonance can be made obvious. Having points far from resonance is important in order to observe the full zero-to pi swing. For measuring phase use a photogate to measure the rotating arm when it is horizontal at its furthest from the cart. At zero phase this should correspond to the point when the cart is the furthest from the motion sensor. Compare the time the gate closes (average value of the gate states) to the time when the cart position is at its maximum. This is the phase in time. To get it into radians:

$$\delta = \omega(t_{gate} - t_{peak})$$

To get proper behavior, t_gate-t_peak should always be positive (except at low frequencies)

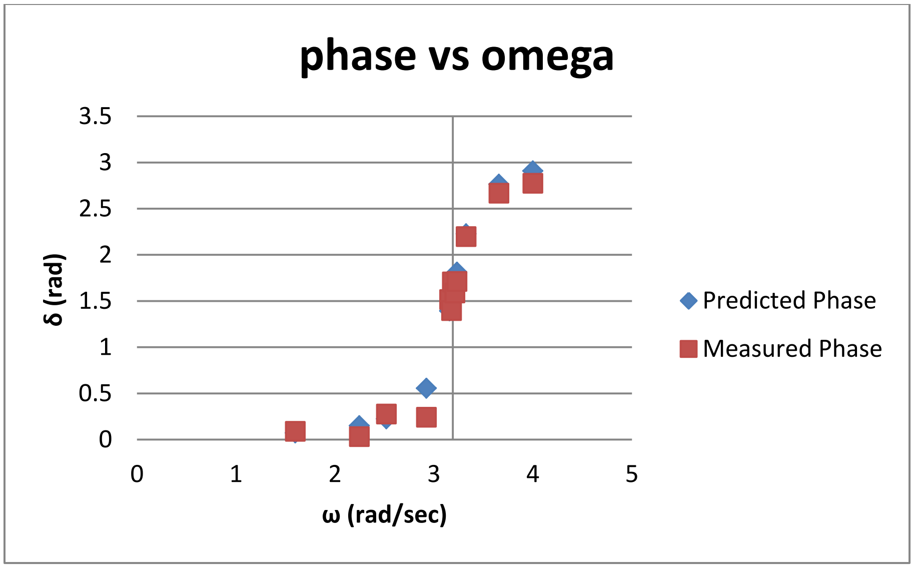


Figure 14: values should vary between 0, and π, with the phase at the peak resonance being $\frac{\pi}{2}$. Sometimes, this gets squashed (so it varies between 0 and 2.4 for example).

Appendix A: Python codes for lab 4

4.2.1 The law of orbits

```
from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
initial=(-5,0,0)
acc=vector(0,0,0)
acc2=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,7,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
0000000)
dt=0.001
maxtsteps=1500
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
                title="y vs x", xtitle="x position",
                ytitle="y position",
                xmin=-5.5, xmax=5, ymin=0, ymax=25,
                foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
    rate(1000)
    r=m2.pos-m1.pos
    acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
    m1.vel=m1.vel+acc*dt
    m1.pos=m1.pos+m1.vel*dt
    m1.trial.append(pos=m1.pos)
    xypoints.append((m1.pos.x,m1.pos.y))
xyplot=gdots(pos=xypoints,color=color.blue)
```

4.2.2 The Law of Areas

```
from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
r2=vector(0,0,0)
dA=0
initial=(-5,0,0)
acc=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,7,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
0000000)
dt=0.001
maxtsteps=1500
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
              title="y vs x", xtitle="x position",
              ytitle="y position",
              xmin=-5.5, xmax=5, ymin=0, ymax=25,
              foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
   rate(1000)
   r2=r
   r=m2.pos-m1.pos
   dA=abs(cross(r,r2))/2
   print "Area from time ",tstep*dt-dt," to ",tstep*dt," = ",dA
   acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
   m1.vel=m1.vel+acc*dt
   m1.pos=m1.pos+m1.vel*dt
   m1.trial.append(pos=m1.pos)
   xypoints.append((m1.pos.x,m1.pos.y))
xyplot=gdots(pos=xypoints,color=color.blue)
```

Sample Output:

```
Area from time  0.001  to  0.002  =  0.0175
Area from time  0.002  to  0.003  =  0.0175
Area from time  0.003  to  0.004  =  0.0175
Area from time  0.004  to  0.005  =  0.0175
Area from time  0.005  to  0.006  =  0.0175
Area from time  0.006  to  0.007  =  0.0175
Area from time  0.007  to  0.008  =  0.0175
Area from time  0.008  to  0.009  =  0.0175
Area from time  0.009  to  0.01  =  0.0175
Area from time  0.01  to  0.011  =  0.0175
Area from time  0.011  to  0.012  =  0.0175
Area from time  0.012  to  0.013  =  0.0175
Area from time  0.013  to  0.014  =  0.0175
Area from time  0.014  to  0.015  =  0.0175
Area from time  0.015  to  0.016  =  0.0175
```

4.2.3 The Law of Periods

```
from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
initial=(-5,0,0)
maxdistance=0
mindistance=100000000
period=0
periodcount=0
acc=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,7,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=100000000000
00)
dt=0.001
maxtsteps=1500
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
                title="y vs x", xtitle="x position",
                ytitle="y position",
                xmin=-5.5, xmax=5, ymin=0, ymax=25,
                foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
    rate(1000)
    r=m2.pos-m1.pos
    if abs(m1.pos - initial) < 1e-2 and tstep>100 :
        period=(period*periodcount+tstep*dt)/(periodcount+1)
        periodcount=periodcount+1
    maxdistance=max(maxdistance,abs(r))
    mindistance=min(mindistance,abs(r))
    acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
    m1.vel=m1.vel+acc*dt
    m1.pos=m1.pos+m1.vel*dt
    m1.trial.append(pos=m1.pos)
    xypoints.append((m1.pos.x,m1.pos.y))
print "Period = ", period
print "Max distance = ",maxdistance
print "Min distance = ",mindistance
semimajoraxis=(maxdistance+mindistance)/2
print "Semi-major axis = ",semimajoraxis
print "Period^2/Semi-major axis^3 = ",(period**2/semimajoraxis**3)
xyplot=gdots(pos=xypoints,color=color.blue)
```

Sample Output:

```
Period =  1.3045
Max distance =  5.0000034971
Min distance =  1.12434686967
Semi-major axis =  3.06217518338
Period^2/Semi-major axis^3 =  0.059264970048
```

4.3.1 Angular Momentum

```
from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
L=0
initial=(-5,0,0)
acc=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,7,0)
m1.trail=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
0000000)
dt=0.001
```

```
maxtsteps=1500
L=arrow(pos=(0,0,0),axis=(0,0,0),shaftwidth=0.01)
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
                  title="y vs x", xtitle="x position",
                  ytitle="y position",
                  xmin=-5.5, xmax=5, ymin=0, ymax=25,
                  foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
    rate(1000)
    r=m2.pos-m1.pos
    L.axis=cross(m1.mass*m1.vel,r)
    print "Angular Momentum at t=",tstep," is ",L.axis
    acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
    m1.vel=m1.vel+acc*dt
    m1.pos=m1.pos+m1.vel*dt
    m1.trail.append(pos=m1.pos)
    xypoints.append((m1.pos.x,m1.pos.y))
xyplot=gdots(pos=xypoints,color=color.blue)
from visual import *
```

```
Sample Output:
Angular Momentum at t= 1  is  <0, 0, -35>
Angular Momentum at t= 2  is  <0, 0, -35>
Angular Momentum at t= 3  is  <0, 0, -35>
Angular Momentum at t= 4  is  <0, 0, -35>
Angular Momentum at t= 5  is  <0, 0, -35>
…
Angular Momentum at t= 1497  is  <0, 0, -35>
Angular Momentum at t= 1498  is  <0, 0, -35>
Angular Momentum at t= 1499  is  <0, 0, -35>
```

4.3.2 Precession of the Perihelion

```
from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
initial=(-5,0,0)
```

```
acc=vector(0,0,0)
acc2=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,5,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
0000000)
dt=0.001
maxtsteps=3500
peri=vector(0,0,0)
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
            title="y vs x", xtitle="x position",
            ytitle="y position",
            xmin=-5.5, xmax=5, ymin=0, ymax=25,
            foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
    rate(1000)
    r=m2.pos-m1.pos
    minusOne=1*(peri)#need to lock in current value
    peri=1*(m1.pos)
    # the exponent in the equation below will be modified for each
situation
    acc=(G*m2.mass/(abs(r)**2.5))*((r)/abs(r))
    m1.vel=m1.vel+acc*dt
    m1.pos=m1.pos+m1.vel*dt
    m1.trial.append(pos=m1.pos)
    plusOne=1*(m1.pos)
    xypoints.append((m1.pos.x,m1.pos.y))
    if abs(plusOne) > abs(peri):
        if abs(minusOne) > abs(peri):
            print "perihelion at",peri,"."
xyplot=gdots(pos=xypoints,color=color.blue)
```

4.3.3 Binary Stars

```
from visual import *
from visual.graph import *
```

```
G=.00000000006673
r=vector(0,0,0)
initial=(-5,0,0)
acc=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=100000000
0000)
m1.vel=vector(0,2,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
000000)
m2.vel=vector(0,-2,0)
m2.trial=curve(color=m2.color)
cm=sphere(pos=(0,0,0),radius=0.1,color=color.red,mass=1)
dt=0.005
maxtsteps=1500
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
          title="y vs x", xtitle="x position",
          ytitle="y position",
          xmin=-5.5, xmax=5, ymin=0, ymax=25,
          foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
  rate(1000)
  r=m2.pos-m1.pos
  cm.pos=(m1.pos+m2.pos)/2
  acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
  acc2=-(G*m1.mass/(abs(r)**2))*((r)/abs(r))
  m1.vel=m1.vel+acc*dt
  m2.vel=m2.vel+acc2*dt
  m1.pos=m1.pos+m1.vel*dt
  m2.pos=m2.pos+m2.vel*dt
  m1.trial.append(pos=m1.pos)
  m2.trial.append(pos=m2.pos)
  xypoints.append((m1.pos.x,m1.pos.y))
xyplot=gdots(pos=xypoints,color=color.blue)
```

4.3.5 Tides

```
#(this only gives the basic outline, the values to match with actual
#moon/earth dynamics are not present in this code)

from visual import *
from visual.graph import *

G=.00000000006673
r=vector(0,0,0)
initial=(-5,0,0)
acc=vector(0,0,0)
acc2=vector(0,0,0)
m1=sphere(pos=initial,radius=0.1,color=color.blue,mass=1)
m1.vel=vector(0,7,0)
m1.trial=curve(color=m1.color)
m2=sphere(pos=(0,0,0),radius=0.1,color=color.green,mass=1000000
0000000)
dt=0.001
maxtsteps=1500
print "simulated time will be ",maxtsteps*dt," seconds."
trajplot=gdisplay(x=0,y=0,width=800,height=400,
                  title="y vs x", xtitle="x position",
                  ytitle="y position",
                  xmin=-5.5, xmax=5, ymin=0, ymax=25,
                  foreground=color.black, background=color.white)
xypoints=[]
for tstep in range(1,maxtsteps):
    rate(10000)
    r=m2.pos-m1.pos
    acc=(G*m2.mass/(abs(r)**2))*((r)/abs(r))
    m1.vel=m1.vel+acc*dt
    m1.pos=m1.pos+m1.vel*dt
    m1.trial.append(pos=m1.pos)
    xypoints.append((m1.pos.x,m1.pos.y))
    #tides stuff
    rotation=2*3.14/50# should give ~30 rotations in one orbit
    #gives a coast rotating around the earth,
    #since earth is at the origin, it also give the vector from earth to it
    coast=vector(0.1*cos(rotation*tstep),0.1*sin(rotation*tstep),0)
    moonR=m1.pos-coast
    if abs(cross(moonR,coast)) < 0.03:
        print "High tide at ", tstep*dt
    if abs(dot(moonR,coast)) < 0.03:
        print "Low tide at", tstep*dt
xyplot=gdots(pos=xypoints,color=color.blue)
```