Lab #7– Energy Accounting in Gravitational Orbits

OBJECTIVES

In this lab you will:

- Use the energy principle to analyze how various energies are related to one another in a system consisting of the Earth and a spacecraft orbiting it.
- Use the energy principle to analyze how various energies are related to one another in a system consisting of just a spacecraft orbiting the Earth.
- Use the energy principle to relate the work and potential energy from the two different systems.
- Generate plots in VPython of the energies and works in the two systems to see if your predictions are correct.
- Investigate how sensitive calculations of energy are on the size of the time step.

PART I: The Orbits of a Craft and a Binary System

In the Moon Voyage lab, we looked at the elliptical orbits that arise from two bodies interacting through the gravitational force. Here, we shall look at the conditions for circular orbits to arise, and then investigate what happens when the two objects interacting through the gravitational force have roughly equal masses.

You will need your Moon Voyage program, as we will be reusing almost all of the code.

1. Importing & Adjusting the Code

Open up your Moon Voyage program from Lab 4, and save it as "Orbits.py". Now since we only want to deal with two objects (the Earth and the craft orbiting it), we need to get rid of everything related to the Moon.

- After saving your Moon Voyage program with a different name, comment out everything dealing with the Moon. Make sure the net force acting on the craft is calculated correctly.

- Comment out all of the arrows.

- Comment out the `scene.center` command.

- Run your program to see the craft in an elliptical orbit around the Earth.

2. Circular Orbits

Recall from lecture that for an object in circular motion, the net force on the object takes the form $\left|\vec{F}_{net,\perp}\right| = \frac{|\vec{p}||\vec{v}|}{R}$, where R is the radius of the "kissing circle".

- Calculate the speed needed for the craft to have a circular orbit around the Earth if the craft is to be 10 Earth radii away from the center of the Earth.

- Give the craft the initial velocity you found in above. Run your program to see if your program is functioning correctly.

1.) What can you say about the directions of the momentum vector of the craft and the net force vector on the craft while the craft is undergoing circular motion?

3. Accuracy

If you use a very large Δt, the calculation is inaccurate, because during that large time interval the force changes a lot, making the momentum update inaccurate, and the momentum changes a lot, making the position update inaccurate. On the other hand, if you use a very small Δt for high accuracy, the program runs very slowly. In computational science, there is a trade-off between accuracy and speed.

How can you tell whether an orbit is accurate? There's a simple test: Make Δt smaller and see whether the motion changes. That is, see whether the orbit changes shape. (Obviously the program will run more slowly.)

- Experiment with your value for Δt (don't change your rate statement), to find a good compromise between accuracy and speed. Cut the step size (Δt) and see whether the motion changes or not. Let the craft go around the Earth five times in this test.
- To see the effects of Δt being too large, increase Δt until the motion is different than it is with small Δt. Maximizing the animation window will help you see any differences.
- In your notes, record the (approximate) biggest value of Δt you can use and still get accurate motion in five trips around the Earth.

4. Letting the Earth Move

In the program so far, you have calculated the force that the Earth exerts on the craft, and used that force to change the momentum and position of the craft. From reciprocity, we know that there is a gravitational force from the craft that acts on the Earth. Here we want to add code to the loop to find this force, and then update the Earth's momentum and position.

- Change Δt back to 100, and adjust your rate statement to 2000.
- Before the loop, but after you've defined the Earth and its mass, give the Earth an initial velocity of (0, 0, 0) m/s and write an expression for the Earth's momentum in terms of its velocity and mass.


```
Earth.v = ??
Earth.p = ??
```

- Add code into your loop to calculate the gravitational force on the Earth from the craft, the net force on the Earth, as well as updating the momentum and position of the Earth. Label the sections of code appropriately

```
#Calculate Force of Gravity: Earth
F_Ec = ??
Fnet_E = ??

#Update Momentum: Earth
deltap_Earth = ??
Earth.p = ??

#Update Position: Earth
deltar_Earth = ??
Earth.pos = ??
```

- Run your program to test the validity of the modeling assumption that the Earth doesn't move significantly when the craft is in orbit. That is, see if the orbit of the craft and the motion of the Earth are significantly different than before.
- Answer the following questions:

2.) Was our initial approximation (that the Earth's motion does not significantly affect the orbit of the craft) valid? How can you tell?

3.) How does the gravitational force on the craft from the Earth compare with the gravitational force on the Earth from the craft?

4.) For a given Δt, how does the change in momentum of the Earth compare with the change in momentum of the craft? **Explain your answer using the momentum principle.**

5.) For the same Δt as in question 4, how does the change in the velocity of the Earth compare with the change in the velocity of the craft? **Explain your answer using the momentum principle.**

5. Binary System

In analyzing objects that interact through the gravitational force, we have only looked at systems where one of the objects was much less massive than the other objects near-by, and thus we could consider all of the other objects to not be moving. We now want to see what happens when two objects with comparable masses interact through the gravitational force. Here you will adjust the mass of the craft to be comparable with that of the Earth, and see what types of motion arise.

- Discuss with your group what you think will happen when you give the craft a mass that is comparable with the Earth's mass.

- Increase the mass of the spacecraft until you can see the Earth move significantly when the program is run. Run your program. Does the system behave as you expected?

- Change the mass of the craft to half the mass of the Earth.

- Play around with various initial velocities for the Earth and the craft to see what kinds of orbits are possible.

If you want, we can make sure the camera always stays centered between the Earth and the craft.

- Add the following code inside the loop towards the end. This code will make sure that the center of the graphics window stays at the "center of mass" of the system. The center of mass is a weighted average of the positions of the objects in the system. We will cover center of mass later this semester.


```
scene.center = (Earth.pos*Earth.m +
craft.pos*craft.m)/(craft.m+Earth.m)
```

- Answer the following questions in your notebook:

6.) How does the gravitational force on the craft from the Earth compare with the gravitational force on the Earth from the craft?

7.) For a given Δt, how does the change in momentum of the Earth compare with the change in momentum of the craft?

8.) For the same Δt as in question 4, how does the change in the velocity of the Earth compare with the change in the velocity of the craft?

- Play around with a few different initial velocities for both the craft and the Earth to see what kinds of orbits are possible. You might want to try starting the Earth with zero initial velocity, and giving the craft an initial velocity of (2506, 2506, 0) m/s.

CHECKPOINT 1: Ask an instructor to check your work.
You can read ahead while you're waiting

PART II: Energy within Gravitational Orbits

1. Preparing the Code

Copy your code from the previous section into a new vPython window.

Set the craft's mass equal to 15e3 kg, and its initial velocity to (0, 1100, 0) m/s.

We will be using the approximation that the Earth does not move significantly while the craft is orbiting, so make sure that any motion of the Earth is commented out.

Make sure the simulation time (simtime) is set to run for 4 days.

Make sure Δt is set to 10 and the rate( ) is set to 1000.

Comment our code that modifies autoscaling (scene.autoscale) and scene centering (scene.center).

Run your program to see that the craft has an elliptical orbit around the Earth. If it does not, fix your program.

2. Using the Craft and Earth as the System

Now that we have the program running correctly, we want to take a look at the energies involved in the orbit, and how they depend on the choice of system. First, we'll choose the craft and the Earth as the system.

2.1 Using the Energy Principle

Before adding code to create plots of the energies, we want to work out how we expect the energies of the system to behave.

Answer the following questions:

- **Starting from the energy principle**, work out (using symbols, no numbers other than 0!) how the change in the kinetic energy of the system (ΔK_{sys}) is related to the change in potential energy of the system (ΔU_{sys}) and the work done on the system (W). **Be sure to state what your system is, and make sure to include the kinetic and rest-mass energy terms for all objects in the system!**

  System:

- What objects do work on this system? What is the total work done on the system?

- During a time interval Δt, what is ΔE_{sys} equal to? Show this using your previous answer, or the energy principle.

- During this time interval, how does ΔK_{sys} compare with ΔK_{craft}?

2.2 Plotting the Energies

Now we want to add plots to our VPython program to see if our predictions are accurate. Immediately following the line of code that reads

```
from visual import *
```

add the line:

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

This line tells VPython to let the program have access to the graphing routines. We now need to create objects that VPython will plot.

To make sure that the axes of our plot are labeled accurately, add the following line before the loop, but after the "Objects" section of your code, to a new section called "graphing objects". This will label the axes of our plots with the units we are using.

```
#graphing objects
gdisplay(xtitle='Seconds',ytitle='Joules', x=500, y=0,
width=800, height=500)
```

Now we need to create three **`curve`** objects for the energies (kinetic, potential, and total) that we wish to plot.

Immediately after the "**`gdisplay`**" line of code, add the following lines:

```
Kgraph = gcurve(color=color.magenta)
Ugraph = gcurve(color=color.yellow)
Etotal = gcurve(color=color.red)
```

Inside the loop, add lines for calculating the kinetic energy of the system, the potential energy of the system, and the total mechanical energy of the system. You should add the lines into the loop after you have updated the position of the craft, immediately before the

time update. Define kinetic energy using the momentum of the craft, we can't use craft.v as we don't update the velocity.

```
#calculate System Energies
pmag = mag(craft.p)  ## magnitude of the craft's momentum
K = ??
U = ??
E = ??
```

NOTE: Here we want you to ignore the rest-mass energies when calculating the total energy of the system (**E**). It is important to remember that rest mass energies do count towards the total system energy, but since they are constant here we are neglecting them to make the plots more readable. What we are calculating here is sometimes called the "total mechanical energy".

Add the following code inside the loop, after you calculate the energies, but before the time is incremented. This will add a point to each of the plots.

```
#add data points to Energy graphs
Kgraph.plot(pos=(t,K))
Ugraph.plot(pos=(t,U))
Etotal.plot(pos=(t,E))
```

Compare the behavior of your plots with your predictions from Section 1. Do they agree? Why is $K \neq -U$? **Look carefully and think about the Energy Principle!**

2.3 Investigating the Accuracy of the Energy Calculations

In your lab on circular orbits, you investigated how the size of the time step Δt can affect the accuracy of the momentum and position update formulas, causing anomalies to occur in the program (such as circular orbits not actually being circular). Energy is a much more sensitive indicator of how "good" Δt is than momentum or position. Here we'll see how changing Δt affects the energy of the system.

Change the time step to 100 seconds. Run your program.

Answer the following question:

1.) Do your graphs still agree with your predictions? Closely examine the total energy curve.

2.) Change the time step back to 10 seconds. Do you still see the anomaly you spotted in question 1?

3.) Why does the anomaly show up where it does?

Change the time step to 1 second and run your program to see that the total energy curve now appears constant.

4.) What can you conclude about the sensitivity of E to Δt?

CHECKPOINT 2: Ask an instructor to check your work.
You can read ahead while you're waiting

3. Using the Craft as the System

Now we are going to analyze the same physical situation (craft orbiting the Earth), but we will consider the craft to be our system, not the craft and the Earth as in Section 2.

3.1 Using the Energy Principle

We need to work out on paper our predictions for how the energies of the system will be related to one another.

Answer the following questions:

- **Starting from the energy principle**, work out (using symbols, no numbers other than 0!) how the change in the kinetic energy of the system (ΔK_{sys}) is related to the work done on the craft (W). **Be sure to state what your system is, and make sure to include the kinetic and rest-mass energy terms for all objects in the system!**

  System:

- Is there any potential energy in this system? If so, what is it? If not, why not?

- During a small time interval Δt when the craft moves a small displacement $\Delta\vec{r}$, what is ΔE_{sys} equal to? Show this starting with the energy principle. Your answer should be expressed in terms of the forces acting on the system.

- During this time interval, how does ΔK_{sys} compare with ΔK_{craft}?

3.2 Plotting the Energies

We need to add a plot for the work done on the craft. This means we have to create another "**`curve`**" object.

Add the following code to your program in the "graphing objects" section of your code, immediately after the "**`Etotal`**" object is created:

```
Wgraph = gcurve(color=color.cyan)
```

At t=0, work has not yet been done on the craft. We need to specify this "initial value" for the work done on the craft.

Add the following code to your program, in the "**initial values**" section of your code.

```
W_oncraft = 0
```

We now need to tell VPython to keep track of the total work done on the craft, by calculating the amount of work done on the craft during each iteration of the loop.

If we know the cumulative work done on a system from $t = 0$ to time t, we can find the total cumulative work done on the system from $t = 0$ to time $t + \Delta t$ by calculating the amount of work done on the system ($W_{interval\ \Delta t}$) during the time interval Δt.

$$W_{from\ t=0\ to\ t+\Delta t} = W_{from\ t=0\ to\ t} + W_{interval\ \Delta t}$$

If the system goes through a displacement $\Delta\vec{r}$ with a net force $\vec{F}_{net}$ acting on it during the interval Δt, we can calculate the additional piece of work ($W_{interval\ \Delta t}$) as:

$$W_{interval\ \Delta t} = \vec{F}_{net} \cdot \Delta\vec{r}$$

We will call the variable $W_{interval\ \Delta t}$, **delta_W.** To calculate this we will have to use the **dot( )** function in VPython which takes the dot product of two vectors. Say we wanted to do **c = a • b**, we would enter **c= dot(a, b)** into our code. Add and complete the following code to your program inside the loop, sometime after the momentum and position have been updated but before the plot commands.

```
#calculate and update work
delta_W = ??  ## work done in this iteration
W_oncraft = ?? ## keeps track of total work on craft
```

Add and complete the following code to the section labeled "add data points to Energy graphs," in order to plot the *cumulative* work done on the craft.

```
Wgraph.plot(pos=(t,??))
```

Edit the kinetic, potential, and total energy calculations so that they are accurate for our choice of system.

Run your program, and compare the behavior of your plots with what you predicted. Do they agree?

Change Δt to 10 seconds again. Does the work plot behave correctly? How can you tell? (Hint: Compare the behavior of the work plot with the behavior of the kinetic energy plot).

4. Using Both System Views

In Sections 2 and 3 of this lab, you have investigated how choosing the system in different ways changes the energies that show up in the system, as well as the work done on the system. Here you will investigate how work in one system may show up as a change in potential energy for a different choice of system.

Answer the following questions:

5.) Using the energy principle **TWICE, for the two different systems**, relate the change in the potential energy of the Earth-craft system (ΔU_{sys}) to the work done on the system consisting of just the craft (W).

6.) Comment out the plot commands for the kinetic and total energies. Edit your program so that it plots the work done on the craft system and the potential energy of the Earth-craft system. Do your plots agree with your answer from question 5?

PART III: Bound Systems

A system that consists of 2 (or more) objects that always stay a finite distance apart is referred to as a "bound" system. For two objects which are interacting gravitationally, the condition that the system consisting of both objects is bound is given by

$$K + U < 0.$$

If you look at the total energy from Section 2, you will see from your plots that $K + U < 0$. An "unbound" orbit consists of a system where

$$K + U > 0.$$

In this case, the two objects will move infinitely far apart, and the craft will keep moving. Using the craft and the Earth as your system (change your energy calculations appropriately), increase the y-component of the craft's initial velocity until you find the speed at which the orbit switches from bound to unbound. That is, find the initial speed such that $K + U = 0$.

- Using the energy principle for this system, analytically calculate what the initial speed should be such that $K + U = 0$ in the space below. Let the initial time be right when the craft is launched, and the final time be when the craft is infinitely far away from the Earth and has come to a stop.

Change your plots so that E, K, U, and W are plotted against the distance between the Earth and the craft, rather than time. Be sure to change the "Seconds" label in your plot to "Meters". Also, change the velocity of the craft back to $\langle 0{,}1100{,}0 \rangle$ m/s.

7.) Consider the Earth-craft system. When the Earth-craft distance is a minimum, is K a maximum or a minimum? What about U?

8.) Consider the Earth-craft system. When the Earth-craft distance is a maximum, is K a maximum or a minimum? What about U?

9.) Are your answers for 7 and 8 in agreement with your findings from the energy principle? Explain.

CHECKPOINT 3: Ask an instructor to check your work.

