

Physics 234: Lab 10

Tuesday, March 29, 2011 / Thursday, March 31, 2011

1. Use the `curl` command to download from the class website everything you'll need for the lab.

```
$ WEBPATH=http://www.ualberta.ca/~kbeach/phys234
$ curl $WEBPATH/docs/Lab10.pdf -O
$ curl $WEBPATH/src/Lab10.tar.gz -O
$ evince Lab10.pdf&
$ tar xzf Lab10.tar.gz
$ cd Lab10
```

2. The program `cannon` simulates two balls, one shot from a cannon (mass m_1 , radius R_1) and the other (mass $m_2 = 1.5m_1$, radius $R_2 = 5R_1$) dropped from above, under freefall conditions:

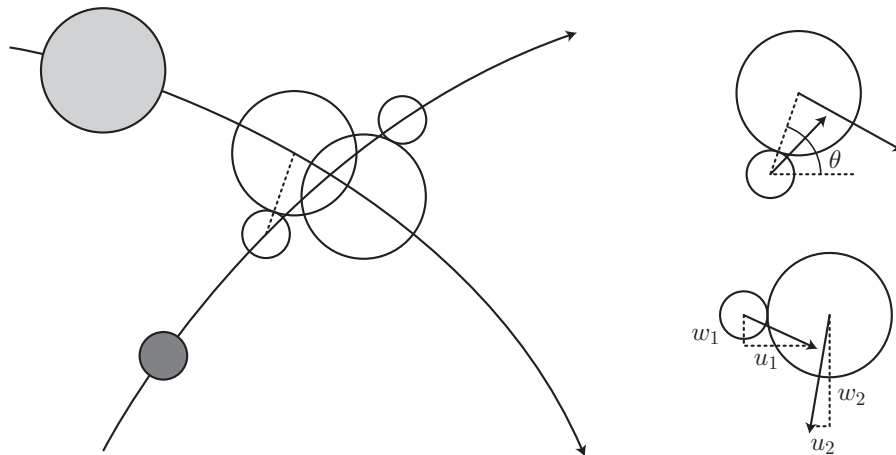
$$x_i(t) = x_i(0) + v_{i,x}(0)t, \quad y_i(t) = y_i(0) + v_{i,y}(0)t - \frac{1}{2}gt^2, \quad (1)$$

$$v_{x,i}(t) = v_{x,i}(0), \quad v_{y,i}(t) = v_{y,i}(0) - gt. \quad (2)$$

To get it working, you'll have to modify the file `cannon.cpp`. First, the positions and velocities should be updated in the function `advance` according to the formula above. Keep in mind that the `ball_t` structure `B` (passed by reference as an argument to the function) has x - and y -coordinates `B.x` and `B.y` and velocity components `B.vx` and `B.vy`.

```
$ make cannon
g++ -c cannon.cpp -O2 -ansi -pedantic -Wall
g++ -c cannon_openGL.cpp -O2 -ansi -pedantic -Wall
g++ -o cannon cannon.o cannon_openGL.o -L/usr/X11R6/lib -lglut -lGLU -lGL -lXmu ...
$ ./cannon
```

3. Read over the function `update` and try to understand what's going on. At each time step, we solve the quadratic equation for $(x_2 - x_1)^2 + (y_2 - y_1)^2 = (R_1 + R_2)^2$ for the possible collision times t_{c1} and t_{c2} . We take the actual collision time to be the smallest such time that lies in the future ($t_c > 0$).



If the collision lies more than Δt in the future, then we simply advance the particles to their $t + \Delta t$ positions. If $0 < t_c < \Delta t$, however, we evolve by t_c , recompute the velocities after collision (by calling `elastic_scatter`) and then evolve by $\Delta t - t_c$.

The new velocities can easily be determined by rotating to a reference frame in which the particles collide head-on along one of the orthogonal axes.

$$\begin{pmatrix} u_i \\ w_i \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} v_{i,x} \\ v_{i,y} \end{pmatrix}$$

The velocities u_1 and u_2 take new values

$$u'_1 = \frac{u_1(m_1 - m_2) + 2m_2u_2}{m_1 + m_2}$$

$$u'_2 = \frac{u_2(m_2 - m_1) + 2m_1u_1}{m_1 + m_2}$$

and are rotated back to the conventional frame:

$$\begin{pmatrix} v'_{i,x} \\ v'_{i,y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} u'_i \\ w_i \end{pmatrix}$$

On careful inspection, you'll notice that the `elastic_scatter` function is specialized to $m_1 = m_2$, in which case the collision velocities are simply swapped: $u'_1 = u_2$ and $u'_2 = u_1$. Implement the general case appropriate for $m_1 \neq m_2$. If you've done this correctly, you should find that energy is properly conserved during collision (to within the limits of floating-point accuracy).

4. Suppose that the cannonball is perfectly sticky and welds itself to its target on collision. In such a situation, the final velocities of both balls must be identical. Conservation of momentum tells us that

$$v'_{1,x} = v'_{2,x} = \frac{m_1v_{1,x} + m_2v_{2,x}}{m_1 + m_2},$$

$$v'_{1,y} = v'_{2,y} = \frac{m_1v_{1,y} + m_2v_{2,y}}{m_1 + m_2}.$$

Implement this in the function `inelastic_scatter`. Comment out the `elastic_scatter` function call in `update`:

```
inelastic_scatter();
//elastic_scatter();
```

Why does conservation of energy break down? Where does the energy go?

5. A test particle of mass m is moving under the gravitational force of a much more massive body ($M \gg m$). It's position $\mathbf{r} = (r_x, r_y)$ and velocity $\mathbf{v} = (v_x, v_y)$ obey $\dot{\mathbf{r}} = \mathbf{v}$ and $\dot{\mathbf{v}} = \mathbf{F}(\mathbf{r})/m$, where \mathbf{F} is the usual $1/r^2$ central force law. For convenience, let's work in units where $m = GM = 1$. The initial conditions are $\mathbf{r}(t = 0) = (-1, 0)$ and $\mathbf{v}(t = 0) = (0.1, 0.7)$.

To obtain an iterative solution, we must discretize the time variable into steps of size Δt and define the on-grid coordinates $\mathbf{r}_n \equiv \mathbf{r}(n\Delta t)$ and $\mathbf{v}_n \equiv \mathbf{v}(n\Delta t)$. In the Euler approximation, the evolution of the ODEs is given by

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_n \Delta t,$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{\Delta t}{m} \mathbf{F}(\mathbf{r}_n) = \mathbf{v}_n - \frac{\Delta t}{m} \frac{GMm \hat{\mathbf{r}}_n}{r_n^2} = \mathbf{v}_n - \Delta t \frac{\mathbf{r}_n}{r_n^3}.$$

The program `orbit.cpp` computes the orbit of the test particle using N Euler steps evenly spaced between $t = 0$ and $t = T = 100$. Compile and run the program with $N = 100000$.

```

$ make orbit
g++ -o orbit orbit.cpp -O2 -ansi -pedantic -Wall
$ ./orbit
Usage: orbit [Euler|EulerCromer] #steps
./orbit Euler 100000
Computing orbit for times up to T = 100 in steps of dt = 0.001
$ gnuplot -persist orbit.gp

```

What does the provided plot script show you?

- How much better is the solution if you increase N by a factor of ten? How much worse is it if you decrease it by a factor of 10?
- The force $F(\mathbf{r}) = -\nabla U(\mathbf{r}) = -\hat{\mathbf{r}}/r^2$ is related to the gravitational potential $U(\mathbf{r}) = -1/r$. Convince yourself that the total energy (kinetic plus potential) of the test particle is $E = \frac{1}{2}v^2 - 1/r = -0.75$. Fill in the body of the function `energy` so that it computes the current total energy. View the energy as a function of time using the provided script. Even for a very large number of time steps ($N = 10^6$), the energy continuously increases.

```

$ make orbit
g++ -o orbit orbit.cpp -O2 -ansi -pedantic -Wall
./orbit Euler 1000000
Computing orbit for times up to T = 100 in steps of dt = 0.0001
$ gnuplot -persist energy.gp

```

- The variables `double rx, ry, vx, vy` are used in the code to store the current values of the test particle's coordinates and velocities. Introduce a second test particle—with variables `Rx, Ry, Vx, Vy`, assigned the same initial conditions as the original test particle—but allow it to evolve (via `update`) by an increment $\Delta t/5$ five times for each conventional time step. Have the coordinates `Rx` and `Ry` of the second test particle written to columns four and five of the file `orbit.dat` and have its energy written to the third column of `energy.dat`.

```

$ make orbit
g++ -o orbit orbit.cpp -O2 -ansi -pedantic -Wall
./orbit Euler 1000000
Computing orbit for times up to T = 100 in steps of dt = 0.0001
$ gnuplot -persist orbit2.gp
$ gnuplot -persist energy2.gp

```

The large discrepancies in behaviour of the two test particles indicates that the Euler solution is not well converged on $[0, T]$.

- Implement the function `EulerCromer_update` according to the following rule.

$$\begin{aligned}\mathbf{r}_{n+1} &= \mathbf{r}_n + \mathbf{v}_{n+1}\Delta t, \\ \mathbf{v}_{n+1} &= \mathbf{v}_n - \Delta t \frac{\mathbf{r}_n}{r_n^3}.\end{aligned}$$

Here, you can get away with performing the operations

$$\begin{aligned}\mathbf{v} &:= \mathbf{v} - \Delta t \frac{\mathbf{r}}{r^3}, \\ \mathbf{r} &:= \mathbf{r} + \mathbf{v}\Delta t.\end{aligned}$$

(Watch out. The order is important.) Now go back and view all the plots again.

```
$ make orbit
g++ -o orbit orbit.cpp -O2 -ansi -pedantic -Wall
./orbit EulerCromer 1000000
Computing orbit for times up to T = 100 in steps of dt = 0.0001
$ gnuplot -persist orbit.gp
$ gnuplot -persist orbit2.gp
$ gnuplot -persist energy2.gp
```

Examine the orbital shape as Δt increases from 0.001 to 0.1. How does the energy behave? You should find that energy conservation is restored—at least on average over each orbital period.

10. Rescale the initial velocity by a factor of 1.95 and view the shape of the orbit. What happens if you double the initial velocity? What is the value of the energy?