Physics 121 – Problem Set # 4

(due Friday, May 10)

1. Griffiths, problem 9.2.

2. Griffiths, problem 9.10.

3. Using the result of the previous problem, compute the radiation pressure on a totally absorbing sphere of radius R in a radiation field of intensity I (in Watts/m^2). Small particles in the gravitational field of the sun are acted on by the sun's gravity and the sun's radiation pressure. (Ignore the solar wind, cosmic rays, *etc.*) Show that the ratio of forces from gravity and radiation pressure is independent of distance. Show that there is a sufficiently small R such that radiation pressure begins to dominate over gravity, and find this value of R for a sphere with density ρ in terms of the acceleration a due to the sun's gravity and the intensity of sunlight I at the earth's radius. Using the value of I from the previous problem, the earth-sun distance of 1.5×10^{11} m, the earth's orbital period of 1 yr $= 3.2 \times 10^7$ sec, and the density $\rho = 2.5$ g/cm^3, find this critical value of R. Radiation pressure is responsible for the fact that the tails of comets point away from the sun. (*cf.* Heald and Marion, problem 5-10)

4. Griffiths, problem 9.33.

5. This problem discusses the numerical solution of the wave equation in one dimension:

$$(\frac{\partial^2}{\partial t^2} - \frac{\partial^2}{\partial x^2}) f(t,x) = 0 \ , \tag{1}$$

where I have set $c = 1$. As we discussed in class, the equation factorizes into

$$(\frac{\partial}{\partial t} - \frac{\partial}{\partial x})(\frac{\partial}{\partial t} + \frac{\partial}{\partial x}) f = 0 \tag{2}$$

The zeros of the first factor are waves that move to the left $f(x+t)$; the zeros of the second factor are waves that move to the right $f(x-t)$. In this problem, we will solve the reduced equation

$$(\frac{\partial}{\partial t} + \frac{\partial}{\partial x}) f(t,x) = 0 \tag{3}$$

numerically. It looks easy, but it is surprisingly subtle.

   (a) We discretize the equation by letting x live on a grid with lattice spacing a, and letting t live on a grid with lattice spacing $a\Delta t$. Then $f_n[j]$ is the value of f at $x = ja$ and $t = na\Delta t$. Show that the following is a simple discretization of

the equation (3), in the sense that we recover the partial differential equation as $a \to 0$:

$$\frac{(f_{n+1}[i] - f_n[i])}{a\Delta t} = -\frac{(f_n[i+1] - f_n[i-1])}{2a} \tag{4}$$

In eq. (4), we can cancel a and solve easily for $f_{n+1}[i]$.

(b) Now implement this solution numerically. As a first step, go to the Physics 121 web site and download the files:

`Wave.html, Wave.java, WaveGUI.java, PhysicsApplet.java` (5)

As written, the applet produces a waveform that shrinks to zero when you click the 'Solve' button. It is also possible to draw in your own initial waveform. The slidebar at the bottom adjusts Δt, which is called `tstep` in the code.

Look at the code of the program `Wave.java`. The routine `solve()` updates $f[i]$ for $i = 1, ..., (Nx - 1)$, where $Nx = 300$ in this case. The function `transfer(f1,f2)` copies `f1` to `f2` in such a way as to preserve periodic boundary conditions.

(c) Modify the code of `Wave.java` to implement the algorithm in part (a). The variable `tstep` has been initialized to represent Δt; use it in your program. When you run the program, you will notice that it is a disaster. Tiny wiggles in the initial condition grow uncontrollably. This is especially noticeable if you draw an initial condition that is not completely smooth. Notice that the instability is lessened if you decrease the time step, but not by much.

(d) Here is alternative way to discretize this equation:

$$\frac{(f_{n+1}[i] - \frac{1}{2}(f_n[i+1] + f_n[i-1]))}{a\Delta t} = -\frac{(f_n[i+1] - f_n[i-1])}{2a} \tag{6}$$

due to Lax. Solve for $f_{n+1}[i]$, and implement this algorithm in the code. Keep the time step less than 1 for the moment. Experiment with different initial waveforms. Notice that small wiggles are smoothed out, but that a large waveform retains its shape and basically evolves correctly. Explain why the small wiggles are smeared out by considering the behavior for small Δt.

(e) To decrease the smearing and to make the whole computation run faster, you can increase the time step. Experiment with this. Notice that the large waveforms are stable when $\Delta t < 1$, but that for $\Delta t > 1$ there is an instability.

(f) Analyze each algorithm by trying a solution

$$f_n[j] = a_n e^{ijk} \tag{7}$$

with j and integer and k the wavenumber. Solve for a_n from an initial condition a_0. Show that the first algorithm is always unstable, and Lax's is unstable when $\Delta t > 1$.

Hand in your code for `Wave.java`, your answers to parts (a) and (e), and some sample screen shots.

To accurately evolve both large- and small-scale wiggles, an even more sophisticated algorithm is needed. The development of such algorithms is complex, and even a black art.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

public class Wave extends WaveGUI{

  void solve(){
    int i;
    while(true) {
     for (int j = 1; j <= 5; j++){
        t++;
        for (i = 1; i < Nx ; i++){
           fnext[i] = (1 - 0.1 * tstep)*f[i];
    /* replace this with code that solves the wave equation */
        }
        transfer(fnext,f);
     }
     refreshPicture();
     if (timetostop) break;
     pause();
    }
  }

  void transfer(double [] f1, double [] f2){
      for (int i = 1; i < Nx ; i++)   f2[i] = f1[i];
      f2[0] = f1[Nx-1];
      f2[Nx] = f1[1];
  }

  void inputWave(){
    for (int n = 0 ; n < Nx; n++){
      double x = (n- 0.1*Nx)/(0.1*Nx);
      f[n] = Math.exp(- 2.0 * x * x);
    }
  }
}
```

Figure 1: The source file `Wave.java`.