Physics 120 – Problem Set # 6

(due Friday, March 1)

1. Griffiths, problem 4.21.

2. Griffiths, problems 4.22.

3. Griffiths, problem 4.28.

4. Griffiths problem 4.36.

5. In this problem, you will generalize the Laplace applet to solve electrostatic problems in two dimensions with a dielectric.

   (a) The first step is to write down a discretization of the electrostatic equations in the presence of a dielectric. In the Laplace applet, we worked with a discretized potential `phi[i][j]`, $i, j = 0 \ldots 100$ and wrote a discrete version of the Laplace equation for these variables. In the presence of dielectric, there still exists a potential and so we can still work with the array `phi[i][j]`. However, it now obeys a different equation, which now involves the dielectric constant. In this problem, ϵ or `epsilon` represents the *dimensionless* dielectric constant, with ϵ_0 divided out.

   The value of ϵ is different at different points: $\epsilon = 1$ in free space, but it has a larger value in the presence of dielectric material. For numerical analysis, these values must also be discretized. For example, you could assign a discrete value of `epsilon[i][j]` to the square bounded by $(i-1, j-1)$, $(i, j-1)$, $(i-1, j)$, (i, j). Then, starting from

   $$\vec{D} = -\epsilon \vec{\nabla} \phi \tag{1}$$

   and discretizing the derivative, you could use the value of ϵ associated with this square to compute the components of the discretized $\vec{D}$ proportional to `(phi[i][j] - phi[i-1][j])` and `(phi[i][j] - phi[i][j-1])`.

   Working in this way, write discrete version of the components of $\vec{D}$ and of the equation

   $$\vec{\nabla} \cdot \vec{D} = 0 \ . \tag{2}$$

   (b) Solve the final equation in part (a) for `phi[i][j]` and define an iterative procedure to compute `phi[i][j]` numerically for a system with Dirichlet boundary conditions.

   (c) Go to the Physics 120 Web site and download the files:

   `Dielectric.html, Dielectric.java, DielectricGUI.java, PhysicsApplet.java, DielectricGUI.jar`

(I apologize that the file `PhysicsApplet.java` is again an updated version different from the previous one.) Compile the applet `Dielectric.java` and run it. This applet manipulates an array `phi[i][j]`, $i, j = 0 \ldots 100$ which represents a an electrostatic potential discretized on a 2-dimensional grid. It also defines an array `State[i][j]`, $i, j = 0 \ldots 100$ which tells whether a given site is cathode, ground, dielectric, or free space. The value of the dielectric constant is fixed by a scrollbar at the bottom of the applet.

You will find that the applet has the following behavior: (1) When you press the button 'Cathode', you will be able to draw in magneta in the box. The corresponding points of an array `phi[i][j]` are set to 100 Volts. (2) When you press 'Ground', you will be able to draw in black in the box. The corresponding points of the array are set to 0 Volts. For definiteness, the black boundary of the box, corresponding to the array elements $i = 0$, $j = 0$, $i = 100$, $j = 100$ are also set to 0 Volts. (3) When you press the button 'Erase', you can erase magneta and black points. (4) When you press the button 'Reset', the drawing of potentials disappears. (5) When you press the button 'Dielectric', you can draw in blue in the box. The corresponding points of are assigned to be dielectric material. (6) When you press the button 'Erase Dielectric', you can erase blue points. (7) When you press the button 'Measure Voltage' and then click in the box, the value of the potential `phi` at that point is displayed. (8) When you press the button 'Compute Energy', a meaningless value is displayed. (11) Finally , when you press the button 'Solve Laplace', the box turns pink or purple but does not do the correct calculation.

Look at the source code, given below, and figure out how the `solve()` method works. The code is very similar to the `Laplace` applet. The new feature is that there is a new possible state, `DielectricState`. The dielectric constant at each site is found by testing for this state in the manner indicated.

(d) Implement the method of part (b) for solving the electrostatic equation by appropriately modifying the `solve()` method.

(e) Try out your solver. Draw some parallel-plate capacitors and see if the resulting potential distributions are reasonable. Here is a nice test: Draw a capacitor half-full of dielectric. Is the spacing of equipotential surfaces correct? Now erase the dielectric. If you now push 'Solve Laplace', the applet will resolve the problem with the same boundary conditions but without the dielectric. Explain the difference between the two solutions.

(f) Test the result of problem 4 with an appropriate geometry of cathode, ground, and dielectric.

(g) Experiment. For what kind of geometry is the set of equipotential surfaces most affected by the presence of dielectric?

(h) Construct a discrete approximation to the energy of the field configuration,

$$E = \int d^3x(\vec{D} \cdot \vec{E})/2 \tag{3}$$

and fill in the function `Energy()` in the applet to compute this energy from the arrays `phi[i][j])` and `epsilon[i][j]`. The value that you return from this function will then appear when you press the button 'Energy/m'. To be concrete, assume that the square is a 10 cm $\times$ 10 cm cross section of a system extended in the third direction so that the discrete spacing of points is $a = 0.1$ cm. Remember again that `epsilon` is the dimensionless dielectric constant. Return the energy/meter in J/m.

(i) Use the Dielectric applet to compute the capacitance of a capacitor which is 3 cm high and has a 2 cm gap. Assume that the capacitor is 20 cm long, and treat this as a very long distance in the third direction. Consider cases in which the capacitor is empty, half-full, and completely full of dielectric. Use $\epsilon = 5$. How does your result compare to the naive results which ignores all edge effects?

Hand in your java code, your answers to (a), (b), and (i), and a few pictures of potentials.

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

public class Dielectric extends DielectricGUI {

  double criterion = 1.0e-4;

  void solve(){
    double maxdiff = 1.0;
    int iteration = 0;
    double e1 = 1.0, e2 = 1.0, e3 = 1.0, e4 = 1.0;
    do {
      maxdiff = 0.0;
      for (int n = 1; n <= 20; n++){
        for (int i = 1; i < Nx; i++){
          for (int j =1 ; j < Ny; j++){
            if (State[i][j] > NormalState) continue;
            double oldphi = phi[i][j];
   /* this code checks whether the queried site has dielectric
      and assigns the value of the dielectric constant accordingly   */
            e1 = (State[i][j] == DielectricState) ?  epsilon : 1.0;
            e2 = (State[i+1][j] == DielectricState) ?  epsilon : 1.0;
            e3 = (State[i][j+1] == DielectricState) ?  epsilon : 1.0;
            e4 = (State[i+1][j+1] == DielectricState) ?  epsilon : 1.0;
   /* put something more sensible here:  */
    double newphi = 33.0;
            phi[i][j] = newphi;
            double delta = Math.abs(newphi-oldphi);
            if (delta > maxdiff) maxdiff = delta;
          }
        }
        iteration++;
    }
    refreshPicture();
    Legend.write("max. diff : "+maxdiff+ "      "+iteration);
    if (timetostop) break;
    } while ( maxdiff > criterion);
  }

  double Energy(){
     return 0.0;
  }
}
```


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