AEP 4380 HW#6: Boundary Value Problems and Relaxation

Gianfranco Grillo

October 26, 2016

1 Background

1.1 Solving Laplace's Equation Using Gauss-Seidel Iteration and SOR

The method of Gauss-Seidel iteration with successive over-relaxation (SOR) is a method in numerical linear algebra used to solve a system of linear equations. It can be used in order to numerically solve boundary value problems in mathematics and physics. A boundary value problem is a differential equation coupled with a set of conditions, or boundaries, upon which the solution will depend. Gauss-Seidel iteration works by elaborating the original problem in the form of a finite difference matrix that is then solved by repeteadly iterating over the matrix and changing the values of each of the matrix components based on the values of each of the components neighbors, starting from an initial guess. After a certain number of iterations, depending on the particular characteristics of the problem, the components in the matrix will converge towards stable values that will correspond to the problem's solution. This algorithm is useful in the field of electrostatics, where it is often necessary to calculate the value of potentials and electric fields throughout a certain region of space, and it might prove difficult or impossible to do so using analytical methods. This homework will use Gauss-Seidel iteration in this context.

The potential $V = \phi(z, r)$ at a certain point in empty space must satisfy the Laplace equation, given by

$$\nabla^2 \phi = 0 \tag{1}$$

In cylindrical coordinates, and assuming azimuthal independence, (1) becomes

$$\nabla^2 \phi = \left(\frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} \right) \phi(z, r) = 0 \tag{2}$$

Using a finite difference grid with $z = ih$ and $r = jh$, where h corresponds to the grid spacing, the components of the finite difference matrix used to solve (2) are given by

$$\phi_{i,j} = \frac{1}{4}[\phi_{i,j+1} + \phi_{i,j-1} + \phi_{i-1,j} + \phi_{i+1,j}] + \frac{1}{8j}[\phi_{i,j+1} - \phi_{i,j-1}] \, ; \, j > 0 \tag{3}$$

$$\phi_{i,j} = \frac{1}{6}[4\phi_{i,j=1} + \phi_{i+1,j=0} + \phi_{i-1,j=0}] \, ; \, j = 0 \tag{4}$$

Having calculated the values of the components of the finite difference matrix, the new matrix after iteration is given by

$$\phi_{i,j}^{new} = \phi_{i,j}^{old} + \omega(\phi_{i,j}^{FD} - \phi_{i,j}^{old}) \tag{5}$$

Where ω corresponds to the relaxation parameter. Successive overrelaxation requires $1 < \omega < 2$.

1.2 Electric Field and Capacitance

Solving the Laplace equation for each point in the grid allows one to calculate the electric field at each point, since the electric field $\vec{E}$ is related to the potential by the equation

$$\vec{E} = -\vec{\nabla} V \tag{6}$$

In particular, for a 2D set of coordinates $(z\,, r)$ like the one in this problem, the magnitude of the components of the electric field are given by

$$E_r = -\frac{\partial V}{\partial r} \tag{7}$$

$$E_z = -\frac{\partial V}{\partial z} \tag{8}$$

In practice, and in particular in the context of this problem, we can approximate the derivatives using finite difference approximations:

$$E_r = -\frac{\phi_{i=m,j+1} - \phi_{i=m,j-1}}{2h} \tag{9}$$

$$E_z = -\frac{\phi_{i+1,j=n} - \phi_{i-1,j=n}}{2h} \tag{10}$$

Finally, it is possible to find the C capacitance of the particular configuration of electrodes given by calculating the total energy stored in the electric field between each of the electrodes and their respective potential differences V_c:

$$C = \frac{\epsilon_0}{V_c} \int \left| \vec{E} \right|^2 dv \tag{11}$$

Where $\epsilon_0 = 8.8542\, picoFarads\, m^{-1}$ and dv is a volume element. In this case, this will be accomplished by adding the magnitudes of the electric fields in between the electrodes, and then multiplying that by 2π, to account for this particular configuration.

2 Results

2.1 Task 1

Figure 1 shows the number of iterations needed for the maximum difference between ϕ^{old} and ϕ^{FD} is less than 2 volts, as a function of the relaxation parameter ω. The minimum occurs at $\omega = 1.85$, so I decided to use a slightly smaller value of ω, 1.84, for the rest of the operations. The code for this task is given in section 4.1.

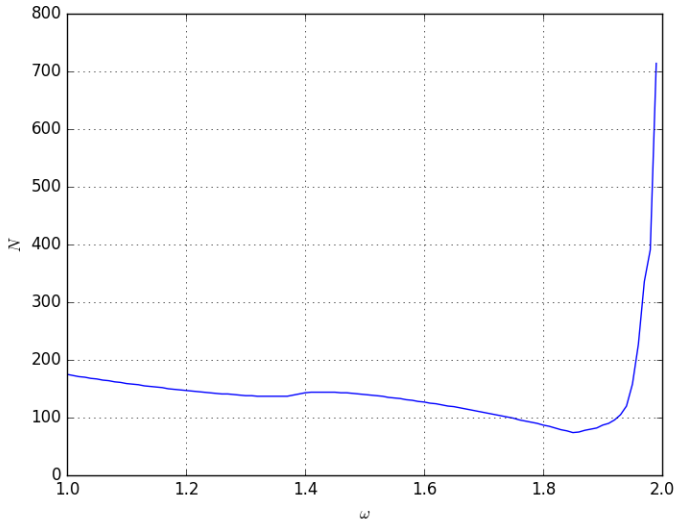

Figure 1: Number of iterations N needed to converge to a 2 volts difference as a function of relaxation parameter ω

2.2 Task 2

Table 1 shows a list of values for the potential V at $r = z = 0$ as a function of h and the tolerance $|V_{max}|$. They are very different from one another simply because the smaller the grid size, the greater the number of points, and the value of the potential divides among a greater number of points (?). The value of the electric field is almost zero in every case, however. I used a grid space of 0.06 and a $|V_{max}|$ of 1.5 volts.[1] The code for this is given in section 4.2.

h	$\lvert V_{max}\rvert$	V
0.05	1.5	0.00501958
0.05	1.6	0.00228572
0.05	1.7	0.00103221
0.05	1.8	0.000461368
0.05	1.9	0.000211358
0.06	1.5	0.201326
0.06	1.6	0.115872
0.06	1.7	0.0663074
0.06	1.8	0.0376722
0.06	1.9	0.0217755
0.1	1.5	26.6209
0.1	1.6	22.8581
0.1	1.7	19.4951
0.1	1.8	16.7167
0.1	1.9	14.1673

Table 1: V at a fixed point for several values of h and $|V_{max}|$

[1] Honestly, I think there is something wrong going on here, but I wasn't able to figure out what I was doing wrong. The following results appear to make more sense, however.

2.3 Task 3

Task 3 involved plotting V and E_z as a function of z along $r = 0$. This is shown in Figure 2. The code for this is given in section 4.3.

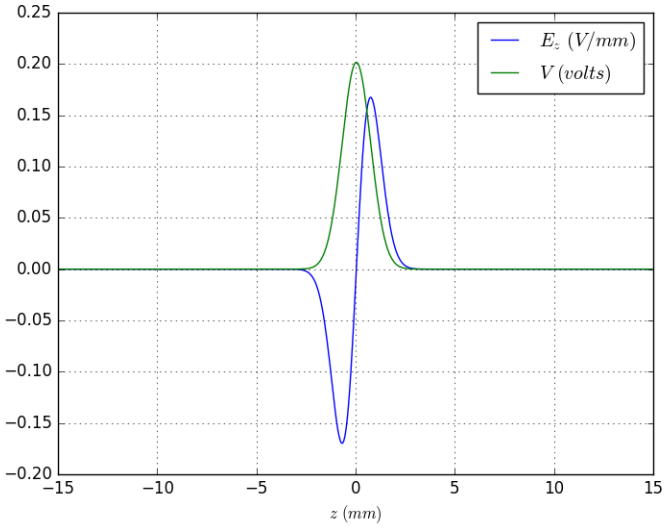

Figure 2: V and E_z as a function of z at $r = 0$

2.4 Task 4

Task 4 involved plotting V, E_r, and E_z along the z axis at $r = 0.75$ mm, with $-10mm < z < 10mm$. The result of this is shown in Figure 3. The code is given in 4.4.

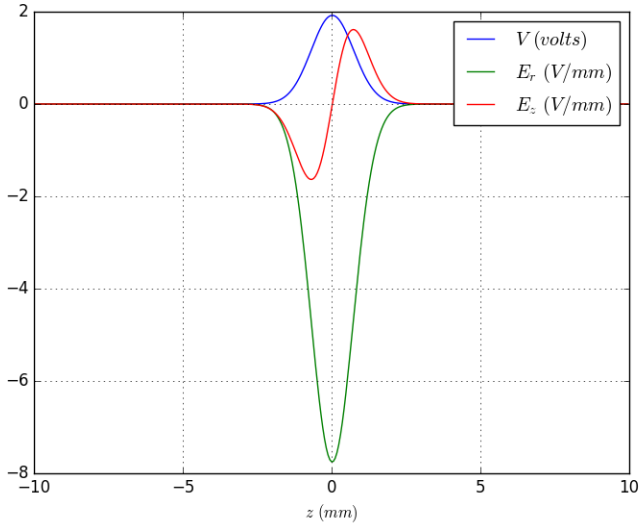

Figure 3: V,E_z, and E_r as a function of z at $r = 0.75mm$

2.5 Task 5

Figure 4 shows a contour plot of the potential as a function of z and r. The code used to produce this is given in 4.5.

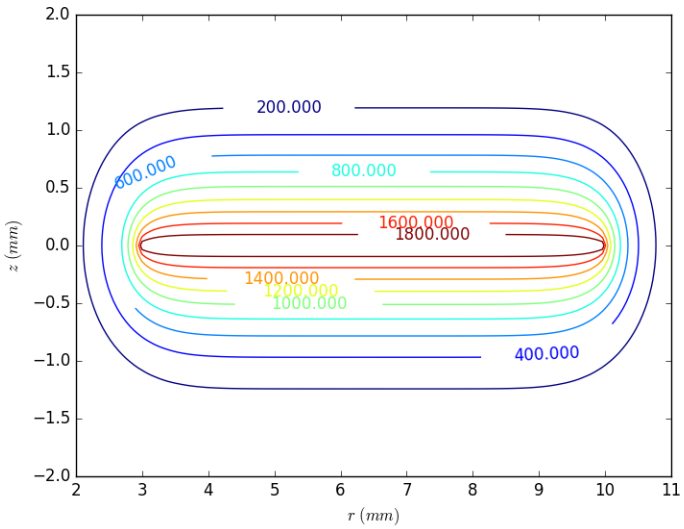

Figure 4: Contour of plot of $V(z, r)$

2.6 Task 6

I calculated the capacitances of both sets of capacitors and got $C_1 = 92.14\, picoFarads$ and $C_2 = 91.79\, picoFarads$. The code used is given in 4.6.

3 Analysis

Most of the results produced make sense, except for the values obtained for the potential as a function of h and $|V_{max}|$, although it is possible that this discrepancy is due to what I suggested earlier, namely that the sum of all of the potentials across each point in space must remain constant, and diminishing the grid spacing implies that the potential values at each point become smaller. I do not know why changing the tolerance produced such drastic changes in the values, however, and I suspect it must be something in my code that I was not able to fix. The relationship between the potential and the electric fields looks good, and the contour plot looks good as well, showing that the potential becomes smaller and smaller the farther away one moves from the electrodes, and it does so in a consistent manner. The values obtained for the capacitances is puzzling, since it seems that the capacitor with the less plate separation (C_1) has a greater capacitance than the one with the less separation (C_2), but its possible that the fringe fields for C_1 are somewhat greater than for C_2 given the respective geometries.

4 Source code

4.1 hw6a.cpp

```
/* AEP 4380 HW#6a
   Boundary value problems and relaxation
   Run on core i7 with g++ 5.4.0 (Ubuntu)
```

```
   Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
        float h = 0.1, rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, tol = 2.0,
            maxdiff;
        int i, j, nite;
        double w;
        void relax(arrayt<float>&, arrayt<char>&, float, float&);
        zlength = zmax/h;
        rlength = rmax/h;
        arrayt<float> oldgridc(zlength, rlength);
        arrayt<float> oldgrid(zlength, rlength);
        arrayt<char> elecloc(zlength, rlength);
        ofstream fp;
    fp.open( "wvsiterations2.dat" );
        for (i = 0; i < zlength; i++) {
                for (j = 0; j < rlength; j++) { // set initial values of potential
                    and flag electrodes
                        z = i*h;
                        r = j*h;
                        if (z == 13.0 && r >= 2.0 && r <= 10.0) {
                                elecloc(i,j) = 1;
                                oldgridc(i,j) = 0.0;
                        }
                        else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
                                elecloc(i,j) = 1;
                                oldgridc(i,j) = 2000.0;
                        }
                        else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
                                elecloc(i,j) = 1;
                                oldgridc(i,j) = 0.0;
                        }
                        else {
                                elecloc(i,j) = 0;
                                oldgridc(i,j) = 0.0;
                        }
                }
        }
        for (w = 1.0; w < 2.0; w = w + 0.01) {
```

```
                nite = 0;
                maxdiff = 2000.0;
                oldgrid = oldgridc;
                while ((maxdiff > tol) && (nite < 10000)) {
                        if (maxdiff > 2000.0) {
                                nite = 10000;
                                break;
                        }
                        relax(oldgrid, elecloc, w, maxdiff);
                        nite++;
                }
                fp << setw(15) << w << setw(15) << nite << endl;
        }
    fp.close();
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff) {
        int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
        float max, diff;
        arrayt<float> fdgrid(zlength, rlength);
        max = 0.0;
        for (i = 1; i < zlength-1; i++) {
                for (j = 0; j < rlength-1; j++) {
                        if (flags(i,j) == 1) {
                                fdgrid(i,j) = oldgrid(i,j);
                        }
                        else if (j == 0) {
                                fdgrid(i,j) = ((4*oldgrid(i,1) + oldgrid(i+1,0) +
                                    oldgrid(i-1,0))/6.0);
                        }
                        else {
                                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j
                                    -1) + oldgrid(i+1,j) + oldgrid(i-1,j)) + (
                                    oldgrid(i,j+1) - oldgrid(i,j-1))/(8*j));
                        }
                        diff = abs(fdgrid(i,j)-oldgrid(i,j));
                        oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
                        if (diff > max) {
                                max = diff;
                        }
                }
        }
        maxdiff = max;
        return;
}
```

4.2 hw6b.cpp

```
/* AEP 4380 HW#6b
  Boundary value problems and relaxation
  Run on core i7 with g++ 5.4.0 (Ubuntu)
```

```
   Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
    float hinit = 0.01, rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, tol,
       maxdiff, w = 1.84, h, finalv0, E;
    int i, j, nite, t;
    zlength = zmax/hinit;
    rlength = rmax/hinit;
    arrayt<float> oldgridc(zlength, rlength);
    arrayt<float> oldgrid(zlength, rlength);
    arrayt<char> elecloc(zlength, rlength);
    float allowedh[3] = {0.05,0.06,0.1};
    void relax(arrayt<float>&, arrayt<char>&, float, float&, float&, float&);
    ofstream fp;
    fp.open( "task2.dat" );
    for (t = 0; t < 3; t++)  {
        finalv0 = 0.0;
        h = allowedh[t];
        zlength = zmax/h;
        rlength = rmax/h;
        oldgridc.resize(zlength, rlength);
        oldgrid.resize(zlength, rlength);
        elecloc.resize(zlength, rlength);
        for (i = 0; i < zlength; i++) for (j = 0; j < rlength; j++) { // set
            initial values of potential and flag electrodes
            z = i*h;
            r = j*h;
            if (z == 13.0 && r >= 2.0 && r <= 10.0) {
                elecloc(i,j) = 1;
                oldgridc(i,j) = 0.0;
            }
            else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
                elecloc(i,j) = 1;
                oldgridc(i,j) = 2000.0;
            }
            else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
                elecloc(i,j) = 1;
                oldgridc(i,j) = 0.0;
            } else {
```

```
            elecloc(i,j) = 0;
            oldgridc(i,j) = 0.0;
        }
    }
    for (tol = 1.5; tol < 2.0; tol = tol + 0.1) {
        oldgrid = oldgridc;
        nite = 0;
        maxdiff = 2000.0;
        while ((maxdiff > tol) && (nite < 10000)) {
            if (maxdiff > 2000.1) {
                nite = 10000;
                break;
            }
            relax(oldgrid, elecloc, w, maxdiff, h, finalv0);
            nite++;
        }
        E = (oldgrid(15+h,0)-oldgrid(15-h,0))/(2*h);
        fp << setw(15) << h << setw(15) << tol << setw(15) << finalv0 << setw
            (15) << E << endl;
    }
}
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff,
    float& h, float& finalv0) {
    int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
    float max, diff;
    arrayt<float> fdgrid(zlength, rlength);
    max = 0.0;
    for (i = 1; i < zlength-1; i++) {
        for (j = 0; j < rlength-1; j++) {
            if (flags(i,j) == 1) {
                fdgrid(i,j) = oldgrid(i,j);
            }
            else if (j == 0) {
                fdgrid(i,j) = (4*oldgrid(i,1) + oldgrid(i+1,0) + oldgrid(i-1,0))
                    /6.0;
            } else {
                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j-1) + oldgrid(i+1,
                    j) +
                                                oldgrid(i-1,j)) + (oldgrid(i,j+1) -
                                                    oldgrid(i,j-1))/(8*j));
            }
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            diff = abs(fdgrid(i,j)-oldgrid(i,j));
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            if (diff > max) {   max = diff; }
        }
    }
    maxdiff = max;
    finalv0 = oldgrid(15.0/h,0);
```

```
    return;
}
```

4.3 hw6c.cpp

```
/* AEP 4380 HW#6c
   Boundary value problems and relaxation
   Run on core i7 with g++ 5.4.0 (Ubuntu)
   Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
    float rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, maxdiff, w = 1.84, h =
        0.06, tol = 1.5, V, E;
    int i, j, nite, t;
    zlength = zmax/h;
    rlength = rmax/h;
    arrayt<float> oldgrid(zlength, rlength);
    arrayt<char> elecloc(zlength, rlength);
    void relax(arrayt<float>&, arrayt<char>&, float, float&);
    ofstream fp;
    fp.open( "task3.dat" ); // open new file for output
    if( fp.fail() ) { // in case of error
        cout << "cannot open file" << endl;
        return( EXIT_SUCCESS );
    }
    for (i = 0; i < zlength; i++) for (j = 0; j < rlength; j++) { // set initial
        values of potential and flag electrodes
        z = i*h;
        r = j*h;
        if (z == 13.0 && r >= 2.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        }
        else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 2000.0;
        }
        else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
```

```
            oldgrid(i,j) = 0.0;
        } else {
            elecloc(i,j) = 0;
            oldgrid(i,j) = 0.0;
        }
    }
    nite = 0;
    maxdiff = 2000.0;
    while ((maxdiff > tol) && (nite < 10000)) {
        if (maxdiff > 2000.1) {
            nite = 10000;
            break;
        }
        relax(oldgrid, elecloc, w, maxdiff);
        nite++;
    }
    for (t = 0; t < zlength; t++) {
        z = t*h;
        V = oldgrid(t,0);
        if (t < (zlength - 1) && t > 0) {
            E = -(oldgrid(t+1,0)-oldgrid(t-1,0))/(2*h);
        }
        fp << setw(15) << (z-15) << setw(15) << V << setw(15) << E << endl;
    }
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff) {
    int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
    float max, diff;
    arrayt<float> fdgrid(zlength, rlength);
    max = 0.0;
    for (i = 1; i < zlength-1; i++) {
        for (j = 0; j < rlength-1; j++) {
            if (flags(i,j) == 1) {
                fdgrid(i,j) = oldgrid(i,j);
            }
            else if (j == 0) {
                fdgrid(i,j) = (4*oldgrid(i,1) + oldgrid(i+1,0) + oldgrid(i-1,0))
                    /6.0;
            } else {
                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j-1) + oldgrid(i+1,
                    j) +
                                                oldgrid(i-1,j)) + (oldgrid(i,j+1) -
                                                    oldgrid(i,j-1))/(8*j));
            }
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            diff = abs(fdgrid(i,j)-oldgrid(i,j));
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            if (diff > max) {   max = diff; }
        }
    }
```

```
    maxdiff = max;
    return;
}
```

4.4 hw6d.cpp

```
/* AEP 4380 HW#6d
  Boundary value problems and relaxation
  Run on core i7 with g++ 5.4.0 (Ubuntu)
  Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
    float rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, maxdiff, w = 1.84, h =
        0.06, tol = 1.5, V, Ez, Er, rloc;
    int i, j, nite, t;
    zlength = zmax/h;
    rlength = rmax/h;
    arrayt<float> oldgrid(zlength, rlength);
    arrayt<char> elecloc(zlength, rlength);
    void relax(arrayt<float>&, arrayt<char>&, float, float&);
    ofstream fp;
    fp.open( "task4.dat" ); // open new file for output
    for (i = 0; i < zlength; i++) for (j = 0; j < rlength; j++) { // set initial
        values of potential and flag electrodes
        z = i*h;
        r = j*h;
        if (z == 13.0 && r >= 2.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        }
        else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 2000.0;
        }
        else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        } else {
            elecloc(i,j) = 0;
```

```
            oldgrid(i,j) = 0.0;
        }
    }
    nite = 0;
    maxdiff = 2000.0;
    while ((maxdiff > tol) && (nite < 10000)) {
        if (maxdiff > 2000.1) {
            nite = 10000;
            break;
        }
        relax(oldgrid, elecloc, w, maxdiff);
        nite++;
    }
    for (t = 5.0/h; t < 25.0/h; t++) {
        z = t*h;
        rloc = 0.75/h;
        V = oldgrid(t, rloc);
        if (t < zlength - 1 && t > 1) {
            Ez = -(oldgrid(t+1,rloc)-oldgrid(t-1,rloc))/(2*h);
            Er = -(oldgrid(t,rloc+1)-oldgrid(t,rloc-1))/(2*h);
        }
        fp << setw(15) << (z-15) << setw(15) << V << setw(15) << Er << setw(15) <<
            Ez << endl;
    }
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff) {
    int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
    float max, diff;
    arrayt<float> fdgrid(zlength, rlength);
    max = 0.0;
    for (i = 1; i < zlength-1; i++) {
        for (j = 0; j < rlength-1; j++) {
            if (flags(i,j) == 1) {
                fdgrid(i,j) = oldgrid(i,j);
            }
            else if (j == 0) {
                fdgrid(i,j) = (4*oldgrid(i,1) + oldgrid(i+1,0) + oldgrid(i-1,0))
                    /6.0;
            } else {
                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j-1) + oldgrid(i+1,
                    j) +
                                                oldgrid(i-1,j)) + (oldgrid(i,j+1) -
                                                    oldgrid(i,j-1))/(8*j));
            }
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            diff = abs(fdgrid(i,j)-oldgrid(i,j));
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            if (diff > max) {   max = diff; }
        }
    }
```

```
    maxdiff = max;
    return;
}
```

4.5 hw6e.cpp

```
/* AEP 4380 HW#6e
  Boundary value problems and relaxation
  Run on core i7 with g++ 5.4.0 (Ubuntu)
  Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
    float rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, maxdiff, w = 1.84, h =
        0.06, tol = 1.5;
    int i, j, nite, t;
    zlength = zmax/h;
    rlength = rmax/h;
    arrayt<float> oldgrid(zlength, rlength);
    arrayt<char> elecloc(zlength, rlength);
    void relax(arrayt<float>&, arrayt<char>&, float, float&);
    ofstream fp, rp, zp;
    rp.open("cplotr.dat");
    zp.open("cplotz.dat");
    fp.open( "cplotmain.dat" );
    for (i = 0; i < zlength; i++) for (j = 0; j < rlength; j++) { // set initial
        values of potential and flag electrodes
        z = i*h;
        r = j*h;
        if (z == 13.0 && r >= 2.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        }
        else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 2000.0;
        }
        else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
```

```
        } else {
            elecloc(i,j) = 0;
            oldgrid(i,j) = 0.0;
        }
    }
    nite = 0;
    maxdiff = 2000.0;
    while ((maxdiff > tol) && (nite < 10000)) {
        if (maxdiff > 2000.1) {
            nite = 10000;
            break;
        }
        relax(oldgrid, elecloc, w, maxdiff);
        nite++;
    }
    for (i = 0; i < zlength; i++) {
        z = i*h;
        for (j = 0; j < rlength; j++) {
            fp << oldgrid(i,j) << " ";
        }
    zp << (z-15) << endl;
    fp << endl;
    }
    for (j = 0; j < rlength; j++) {
        r = j*h;
        rp << r << endl;
    }
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff) {
    int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
    float max, diff;
    arrayt<float> fdgrid(zlength, rlength);
    max = 0.0;
    for (i = 1; i < zlength-1; i++) {
        for (j = 0; j < rlength-1; j++) {
            if (flags(i,j) == 1) {
                fdgrid(i,j) = oldgrid(i,j);
            }
            else if (j == 0) {
                fdgrid(i,j) = (4*oldgrid(i,1) + oldgrid(i+1,0) + oldgrid(i-1,0))
                    /6.0;
            } else {
                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j-1) + oldgrid(i+1,
                    j) +
                                                oldgrid(i-1,j)) + (oldgrid(i,j+1) -
                                                    oldgrid(i,j-1))/(8*j));
            }
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            diff = abs(fdgrid(i,j)-oldgrid(i,j));
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
```

```
            if (diff > max) {    max = diff; }
        }
    }
    maxdiff = max;
    return;
}
```

4.6 hw6f.cpp

```
/* AEP 4380 HW#6f
  Boundary value problems and relaxation
  Run on core i7 with g++ 5.4.0 (Ubuntu)
  Gianfranco Grillo 10/25/2016
*/

#include <cstdio>
#include <cstdlib>
#include <cmath>

#include <iostream> // stream IO
#include <fstream> // stream file IO
#include <iomanip> // to format the output
#define ARRAYT_BOUNDS_CHECK
#include "arrayt.hpp"

using namespace std;

int main() {
    float rmax = 15.0, zmax = 30.0, z, r, zlength, rlength, maxdiff = 2000.0, w =
       1.84, h = 0.06, tol = 1.5, Er, Ez, Esqr, eps = 8.8543e-3, totalEsqr, C1, C2;
    int i, j, nite, t;
    zlength = zmax/h;
    rlength = rmax/h;
    arrayt<float> oldgrid(zlength, rlength);
    arrayt<char> elecloc(zlength, rlength);
    void relax(arrayt<float>&, arrayt<char>&, float, float&);
    ofstream fp, rp, zp;
    rp.open("cplotr.dat");
    zp.open("cplotz.dat");
    fp.open( "cplotmain.dat" );
    for (i = 0; i < zlength; i++) for (j = 0; j < rlength; j++) { // set initial
       values of potential and flag electrodes
        z = i*h;
        r = j*h;
        if (z == 13.0 && r >= 2.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        }
        else if (z == 15.0 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 2000.0;
        }
```

```
        else if (z >= 16.5 && z <= 18 && r >= 3.0 && r <= 10.0) {
            elecloc(i,j) = 1;
            oldgrid(i,j) = 0.0;
        } else {
            elecloc(i,j) = 0;
            oldgrid(i,j) = 0.0;
        }
    }
    nite = 0;
    maxdiff = 2000.0;
    while ((maxdiff > tol) && (nite < 10000)) {
        if (maxdiff > 2000.1) {
            nite = 10000;
            break;
        }
        relax(oldgrid, elecloc, w, maxdiff);
        nite++;
    }
    totalEsqr = 0.0;
    for (i = (13.0+h)/h; i < 15.0/h; i++) for (j=1; j < rlength-1; j++) {
        z = i*h;
        r = j*h;
        Ez = (oldgrid(i+1,j)-oldgrid(i-1,j))/(2*h);
        Er = (oldgrid(i,j+1)-oldgrid(i,j-1))/(2*h);
        Esqr = Ez*Ez+Er*Er;
        totalEsqr = totalEsqr + Esqr;
    }
    C1 = (eps*totalEsqr*2*3.1415)/(2000*2000);
    cout << C1 << endl;
    totalEsqr = 0.0;
    for (i = (15.0+h)/h; i < 16.5/h; i++) for (j=1; j < rlength-1; j++) {
        z = i*h;
        r = j*h;
        Ez = (oldgrid(i+1,j)-oldgrid(i-1,j))/(2*h);
        Er = (oldgrid(i,j+1)-oldgrid(i,j-1))/(2*h);
        Esqr = Ez*Ez+Er*Er;
        totalEsqr = totalEsqr + Esqr;
    }
    C2 = (eps*totalEsqr*2*3.1415)/(2000*2000);
    cout << C2 << endl;
    return( EXIT_SUCCESS );
}

void relax(arrayt<float>& oldgrid, arrayt<char> &flags, float w, float& maxdiff) {
    int zlength = oldgrid.n1(), rlength = oldgrid.n2(), i, j, n;
    float max, diff;
    arrayt<float> fdgrid(zlength, rlength);
    max = 0.0;
    for (i = 1; i < zlength-1; i++) {
        for (j = 0; j < rlength-1; j++) {
            if (flags(i,j) == 1) {
                fdgrid(i,j) = oldgrid(i,j);
```

```
            }
            else if (j == 0) {
                fdgrid(i,j) = (4*oldgrid(i,1) + oldgrid(i+1,0) + oldgrid(i-1,0))
                    /6.0;
            } else {
                fdgrid(i,j) = (0.25*(oldgrid(i,j+1) + oldgrid(i,j-1) + oldgrid(i+1,
                    j) +
                                                oldgrid(i-1,j)) + (oldgrid(i,j+1) -
                                                    oldgrid(i,j-1))/(8*j));
            }
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            diff = abs(fdgrid(i,j)-oldgrid(i,j));
            oldgrid(i,j) = oldgrid(i,j) + w*(fdgrid(i,j)-oldgrid(i,j));
            if (diff > max) {    max = diff; }
        }
    }
    }
    maxdiff = max;
    return;
}
```

5 References

This homework was done using material obtained from the lectures.