PHYS580 Lab07 Report

Yicheng Feng
PUID: 0030193826

October 9, 2019

Workflow: I use the Linux system, and code C++ in terminals. For visualization, I use the C++ package – ROOT (made by CERN) to make plots. At the end, the reports are written in LaTeX.

The codes for this lab are written as the following files:

- `capacitor_2d.h` and `capacitor_2d.cxx` for the class `Capacitor2D` to solve general ordinary differential equation sets.
- `lab7.cxx` for the main function to make plots.

To each problem of this lab report, I will attach the relevant parts of the code. If you want to check the validation of my code, you need to download the whole code from the link `https://github.com/YichengFeng/phys580/tree/master/lab7`.

(1) Use the starter programs `capacitor_Jacobi.m` and `capacitor_update_Jacobi.m` (or your equivalent programs) to solve the Laplace equation for a parallel-plate capacitor using the Jacobi relaxation algorithm. In the z direction, the plates are assumed to extend from $z = -\infty$ to $+\infty$ (translational symmetry), so the problem is only two-dimensional. Calculate and plot the potential surfaces, equipotential contours and the electric field (cf. pp.140- 141 in the textbook). How many iterations (sweeps through the lattice) are required as a function of the mesh size to reach a given level of accuracy? At fixed mesh size, how does the total number of iterations needed depend on the desired accuracy? Compare your convergence results with the theoretical expectations.

Physics explanation:

We use the Jacobi relaxation to solve this problem numerically. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The potential of the boundary is set to 0, and the potential of the left plate is set to 1 while the right plate -1.

To explore the dependence of the iteration steps on the set accuracy, we fix the grids squares with edge $d = 0.05L$, and change the upper limit of average error from $acc = 10^{-4}$ to $acc = 10^{-9}$. To save space, we will only plot the results from $acc = 10^{-4}$ to $acc = 10^{-6}$ (Fig. 1 to Fig. 3). It seems that the number of iteration is proportional to the logarithm of accuracy ($n \propto \log(acc)$).

acc	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
number of iteration steps	66	104	165	240	318	397

To explore the dependence of the iteration steps on the grid size, we fix the upper limit of average error to $acc = 10^{-6}$, and change the square grid edge d from $0.05L$ to $0.005L$. To save space, we will only plot the results with $d = 0.05L$, $d = 0.02L$, and $d = 0.01L$ (Fig. 3 to Fig. 5). It seems that the number of iteration is proportional to the number of grids ($n \propto 1/d^2$).

d/L	0.05	0.04	0.02	0.01	0.005
number of grids	400	625	1500	10000	40000
number of iteration steps	165	224	610	2060	6426

For the comparison to the theoratical expectation, we don't get the exact theoratical results, but use some qualitative theoratical expectations.

The results should have symmetric about $x = 0$. However, when the d is large, we can see the difference between the left and right halves. This is because the grids are discrete and the number of them is small. When d is small, the symmetry is there, which is as expected.

At the middle of the plates, the electric field should be close to constant. We can see from the middle plots that the contours there are almost equally separated, and the right plots has the electric field has direction towards right and almost equal in magnitude.

Plots:

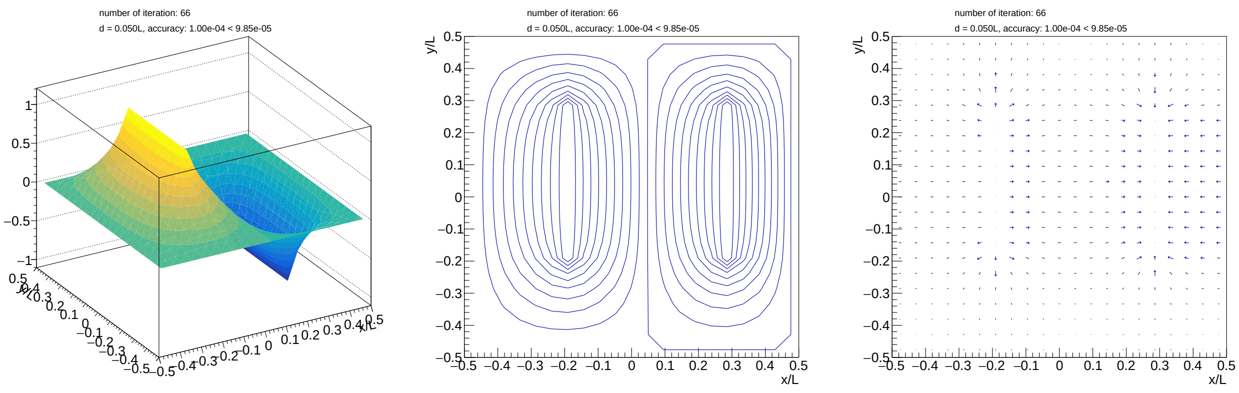

Figure 1: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. The upper limit of average error is $acc = 10^{-4}$.

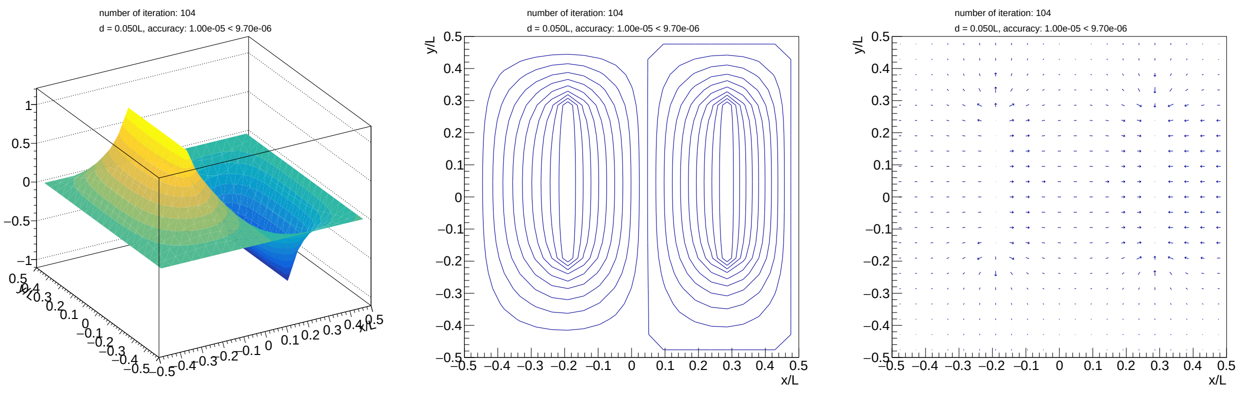

Figure 2: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. The upper limit of average error is $acc = 10^{-5}$.

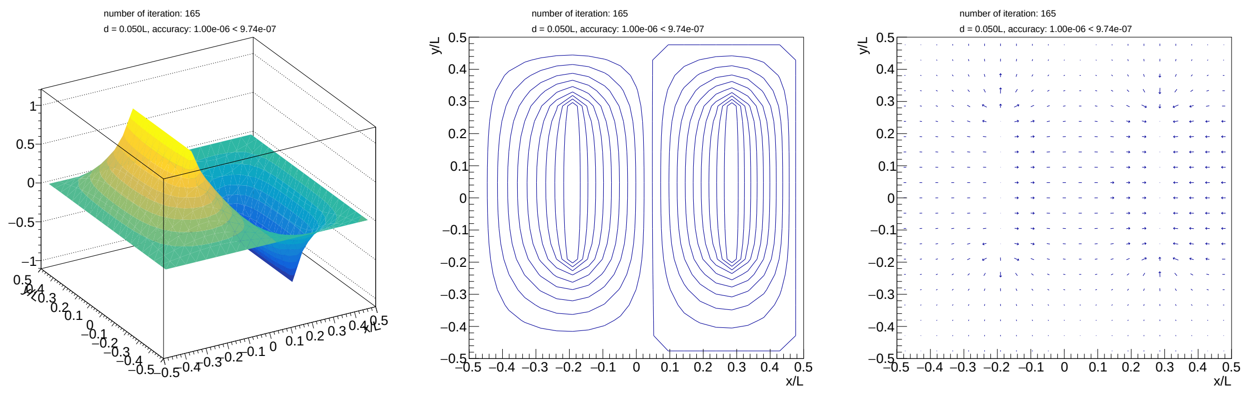

Figure 3: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. The upper limit of average error is $acc = 10^{-6}$.

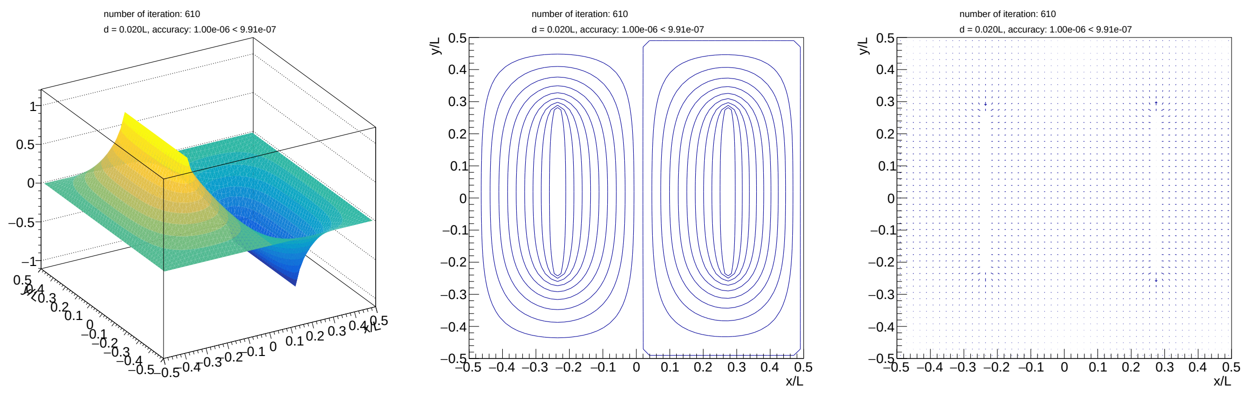

Figure 4: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.02L$. The upper limit of average error is $acc = 10^{-6}$.

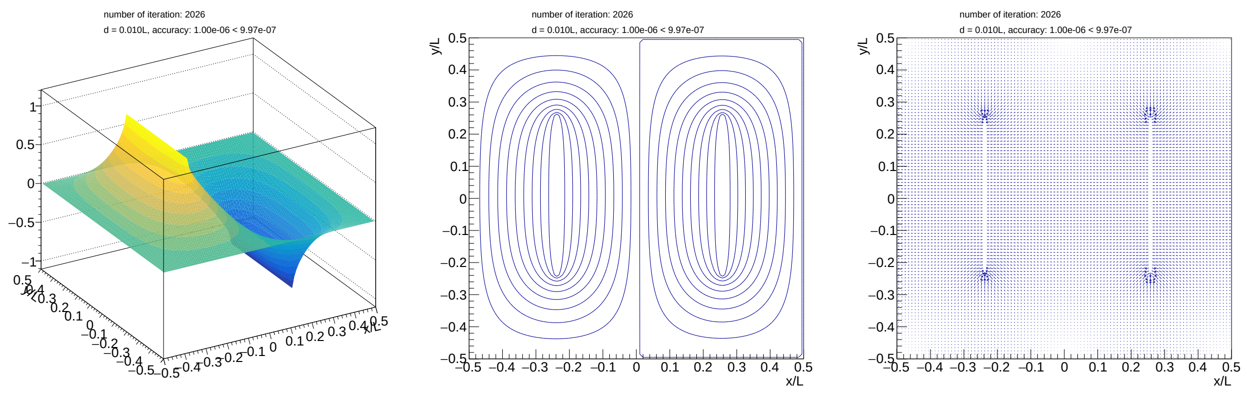

Figure 5: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.01L$. The upper limit of average error is $acc = 10^{-6}$.

Relevant code:

For the Jacobi relaxation

```
//----------------------------------------------------------------------//

void Capacitor2D::cal_once_Jacobi() {

        if(!check()) return;

        vector< vector<double> > Vnew = _V;
        _tmp_acc = 0;

        for(int j=1; j<_N-1; j++) {
                for(int i=1; i<_N-1; i++) {
                        if(_V[i][j]==1 || _V[i][j]==-1) continue;
                        Vnew[i][j] = (_V[i-1][j] + _V[i+1][j] + _V[i][j-1] + _V[i][j+1])*0.25;
                        Vnew[i][j] += _rho[i][j];
                        _tmp_acc += fabs(Vnew[i][j] - _V[i][j]);
                }
        }

        _tmp_acc = _tmp_acc/_N/_N;

        _V = Vnew;

        _n_iter ++;
}

//----------------------------------------------------------------------//
```

For the scan of different d and acc

```
//----------------------------------------------------------------------//

void capacitor_2d_plot(int alg, double d, double acc) {

        TString str_alg[2] = {"Jacobi", "GaussSeidel"};

        TString str_d = Form("d%.4d", int(d*1000));
        TString str_acc = Form("acc%.1d", int(fabs(log10(acc))));

        Capacitor2D c2d(0.5, 0.5, d, acc);
        c2d.set_alg(alg);
        c2d.cal();
        vector< vector<double> > V = c2d.get_V();
        int nx = V[0].size();
        int ny = V.size();
        double tmp_acc = c2d.get_tmp_acc();
        int n_iter = c2d.get_n_iter();

        cout << tmp_acc << " " << nx << " " << ny << " " << n_iter << endl;

        TH2F *h2_V = new TH2F("capacitor2d", "capacitor2d", nx,-0.5,0.5, ny,-0.5,0.5);
        h2_V->GetXaxis()->SetTitle("x/L");
        h2_V->GetYaxis()->SetTitle("y/L");
        TH2F *h2_minus_V = new TH2F("capacitor2d_minus", "capacitor2d_minus",
        nx,-0.5,0.5, ny,-0.5,0.5);
        h2_minus_V->GetXaxis()->SetTitle("x/L");
        h2_minus_V->GetYaxis()->SetTitle("y/L");

        for(int j=0; j<ny; j++) {
                for(int i=0; i<nx; i++) {
                        h2_V->SetBinContent(i+1, j+1, V[j][i]);
                        h2_minus_V->SetBinContent(i+1, j+1, -V[j][i]);
                }
        }

        TString str_adj = str_alg[alg] + "_" + str_d + "_" + str_acc;
        TString str_tmp;

        TLegend *l_iter = new TLegend(0.1,0.9,0.7,0.98);
        l_iter->SetFillStyle(0);
        l_iter->SetBorderSize(0);
        l_iter->AddEntry((TObject*)0, Form("number of iteration: %d",n_iter), "");
        l_iter->AddEntry((TObject*)0, Form("d = %.3fL, accuracy: %.2e < %.2e",d,acc,tmp_acc), "");

        // plot codes: mesh
        str_tmp = "V_mesh_" + str_adj;
        TCanvas *c_V_mesh = new TCanvas("c_"+str_tmp, "c_"+str_tmp, 600,600);
        h2_V->Draw("SURF2");
        l_iter->Draw("same");
        c_V_mesh->SaveAs("./plot/"+str_tmp+".pdf");

        // plot codes: contour
        str_tmp = "V_contour_" + str_adj;
        TCanvas *c_V_contour = new TCanvas("c_"+str_tmp, "c_"+str_tmp, 600,600);
        h2_V->Draw("CONT3");
        l_iter->Draw("same");
        c_V_contour->SaveAs("./plot/"+str_tmp+".pdf");

        // plot codes: gradient
        str_tmp = "E_field_" + str_adj;
        TCanvas *c_E_field = new TCanvas("c_"+str_tmp, "c_"+str_tmp, 600,600);
        h2_minus_V->Draw("ARR");
        l_iter->Draw("same");
        c_E_field->SaveAs("./plot/"+str_tmp+".pdf");
}

//------------------------------------------------------------------------//
```

(2) Implement the Gauss-Seidel method to solve the same capacitor problem as in (1). Calculate the potential and analyze any differences in convergence compared to what you found for the Jacobi algorithm in (1).

Physics explanation:

The Gauss-Seidel relaxation updates the potential in-site timely. The number of iteration is usually smaller than the Jacobi relaxation.

We use the Gauss-Seidel relaxation to solve this problem numerically. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The potential of the boundary is set to 0, and the potential of the left plate is set to 1 while the right plate -1.

To explore the dependence of the iteration steps on the set accuracy, we fix the grids squares with edge $d = 0.05L$, and change the upper limit of average error from $acc = 10^{-4}$ to $acc = 10^{-6}$ (Fig. 6 to Fig. 8). It seems that the number of iteration is proportional to the logarithm of accuracy ($n \propto \log(acc)$). And the number of iteration is indeed smaller than the Jacobi relaxation.

acc	10^{-4}	10^{-5}	10^{-6}
number of iteration steps	49	86	126

To explore the dependence of the iteration steps on the grid size, we fix the upper limit of average error to $acc = 10^{-6}$, and change the square grid edge d from $0.05L$ to $0.01L$. (Fig. 8 to Fig. 10). It seems that the number of iteration is proportional to the number of grids ($n \propto 1/d^2$). And the number of iteration is indeed smaller than the Jacobi relaxation.

d/L	0.05	0.04	0.02
number of grids	400	625	1500
number of iteration steps	126	528	1243

The results here seem a little more unsymmetric than the Jacobi relaxation.

Plots:

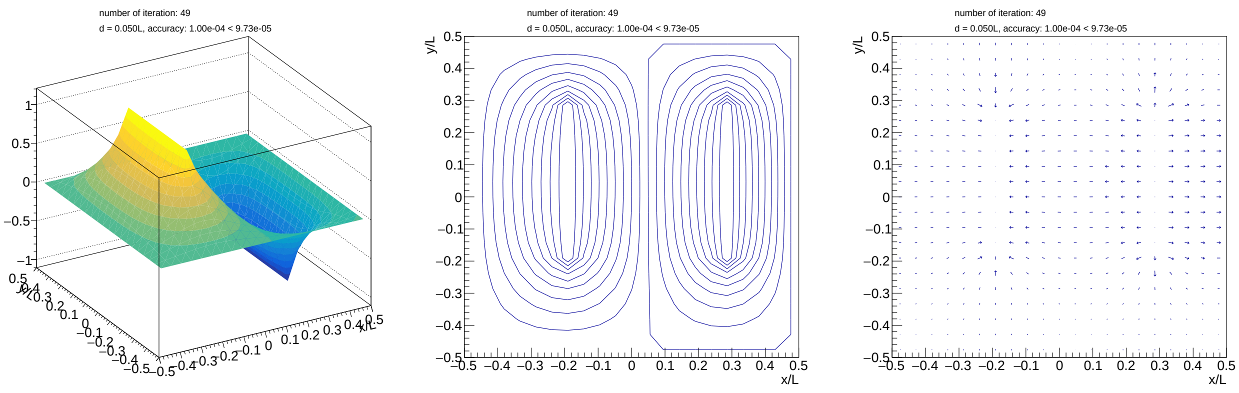

Figure 6: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Gauss-Seidel relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. Set average error is $acc = 10^{-4}$.

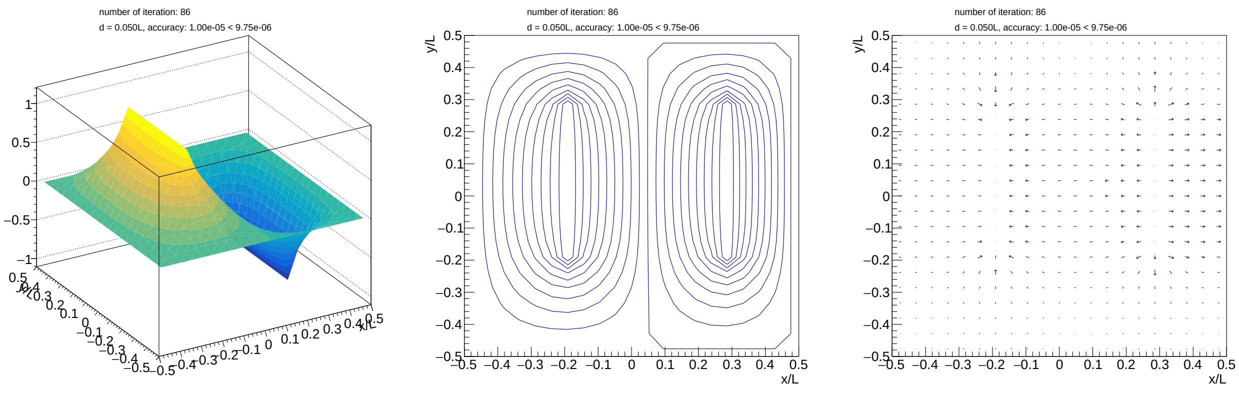

Figure 7: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Gauss-Seidel relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. Set average error is $acc = 10^{-5}$.

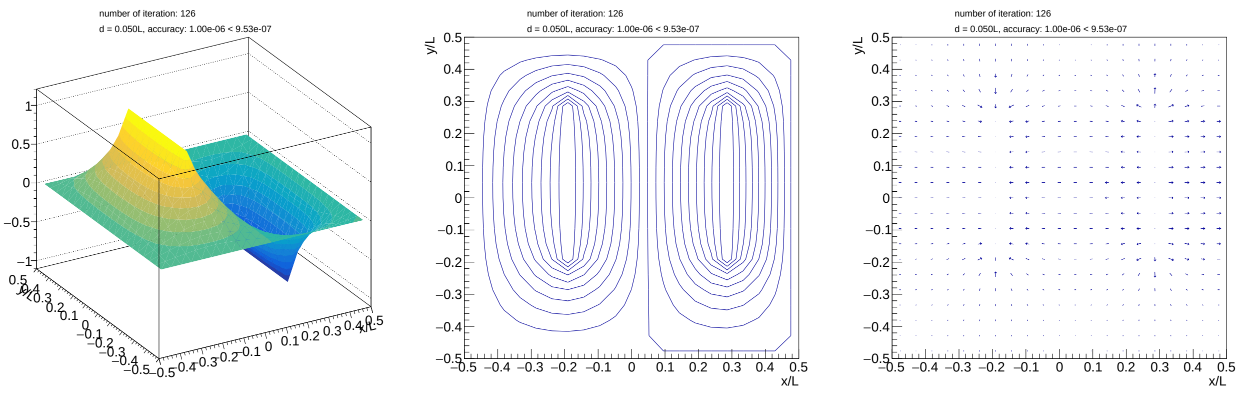

Figure 8: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Gauss-Seidel relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.05L$. Set average error is $acc = 10^{-6}$.

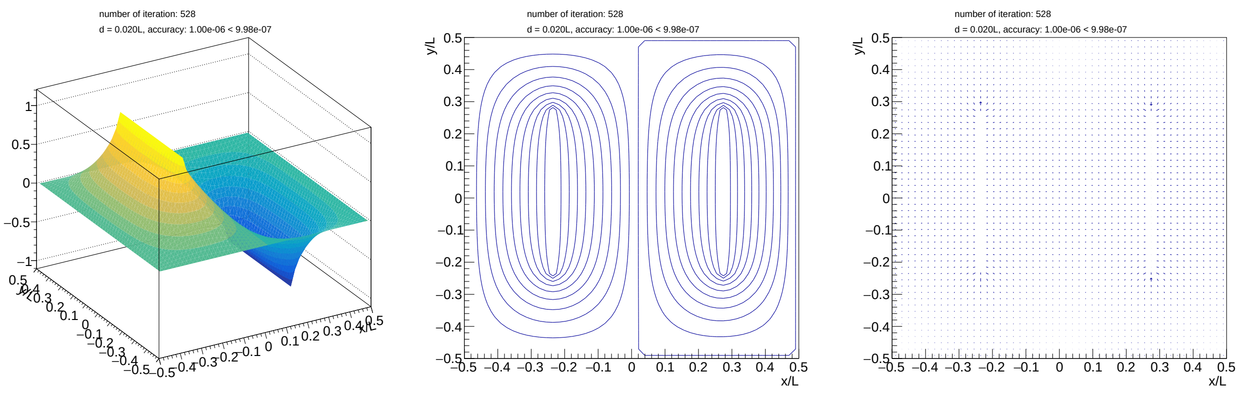

Figure 9: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Gauss-Seidel relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.02L$. Set average error is $acc = 10^{-6}$.

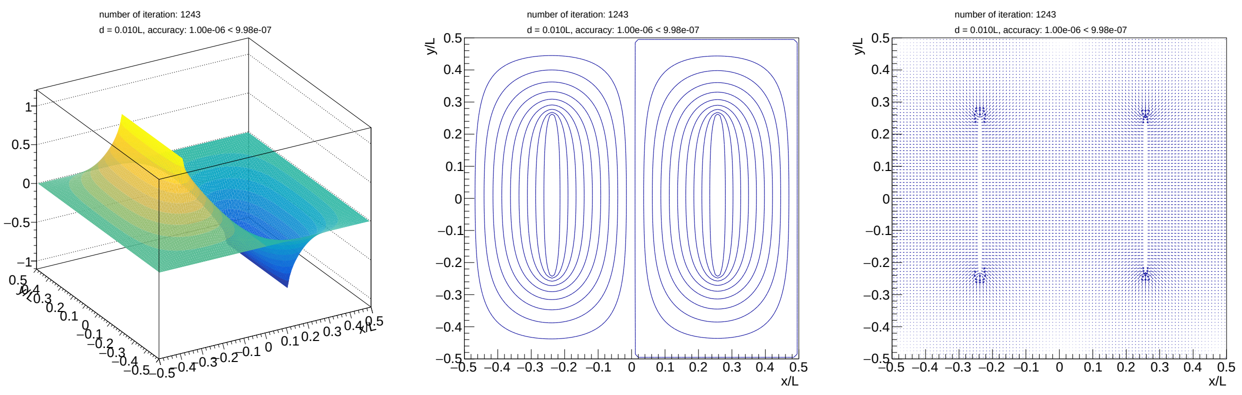

Figure 10: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Gauss-Seidel relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.01L$. Set average error is $acc = 10^{-6}$.

Relevant code:

For the Gauss-Seidel relaxation

```
//---------------------------------------------------------------------//

void Capacitor2D::cal_once_GaussSeidel() {

        if(!check()) return;

        vector< vector<double> > Vold = _V;
        _tmp_acc = 0;

        for(int j=1; j<_N-1; j++) {
                for(int i=1; i<_N-1; i++) {
                        if(_V[i][j]==1 || _V[i][j]==-1) continue;
                        _V[i][j] = (_V[i-1][j] + _V[i+1][j] + _V[i][j-1] + _V[i][j+1])*0.25;
                        _V[i][j] += _rho[i][j];
                        _tmp_acc += fabs(Vold[i][j] - _V[i][j]);
                }
        }

        _tmp_acc = _tmp_acc/_N/_N;

        _n_iter ++;
}

//---------------------------------------------------------------------//
```

(3) Numerically solve the Poisson equation for the capacitor problem in (1) by introducing a term $-\rho/\varepsilon_0$ on the right hand side of the two-dimensional Laplace equation, at your favorite location away from the capacitor and the boundaries. Calculate the electric potentials and fields in this case using any of the three methods discussed so far (see Section 5.2 for how to include charge in the calculation). Compare with theoretical expectations in limiting cases where that is possible (where you can). Explain what the constant term you added corresponds to physically in this 2D calculation; is this a true point charge?

Physics explanation:

We set the charge density equally distributed in the area $-0.05L < x < 0.05L, -0.05L < y < 0.05L$, the total charge is set $Q/\varepsilon_0 = 1$ per length.

We use the Jacobi relaxation to solve this problem numerically. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The potential of the boundary is set to 0, and the potential of the left plate is set to 1 while the right plate -1. The results are shown in Fig. 11.

We also try the periodic boundary conditions with Jacobi relaxation. The plate widths are $a = 1.0L$, and the distance between them is $b = 0.5L$. The potential of the boundary is set periodic, and the potential of the left plate is set to 1 while the right plate -1. The results are shown in Fig. 12.

We compare the periodic results with the theoretical expectation. Out of the plates, the electric field should be constant:

$$\begin{aligned} V(x, y) =& 4(x + 0.5L), \quad -0.5L < x < -0.25L, \\ V(x, y) =& 4(x - 0.5L), \quad 0.25L < x < 0.5L, \end{aligned} \tag{1}$$

which could be seen in all the three plots in Fig. 12.

For the middle input charge density, we can treat it as a 2D point charge and use electric image method to get the total potential between the two plates.

$$V(x, y) = -4x + C - \frac{1}{4\pi} \sum_{k=-\infty}^{+\infty} (-1)^k \ln \left[\frac{(x - kL/2)^2 + y^2}{L^2/16} \right] \tag{2}$$

where C is a constant. We don't spend further time into this, but the numerical results seem consistent with theoretical expectations.

For the constant term, it means the charge density. This is not a point charge. It is actually a infinitely long string perpendicular to the plane. The charge added is actually the charge per unit length.

Plots:

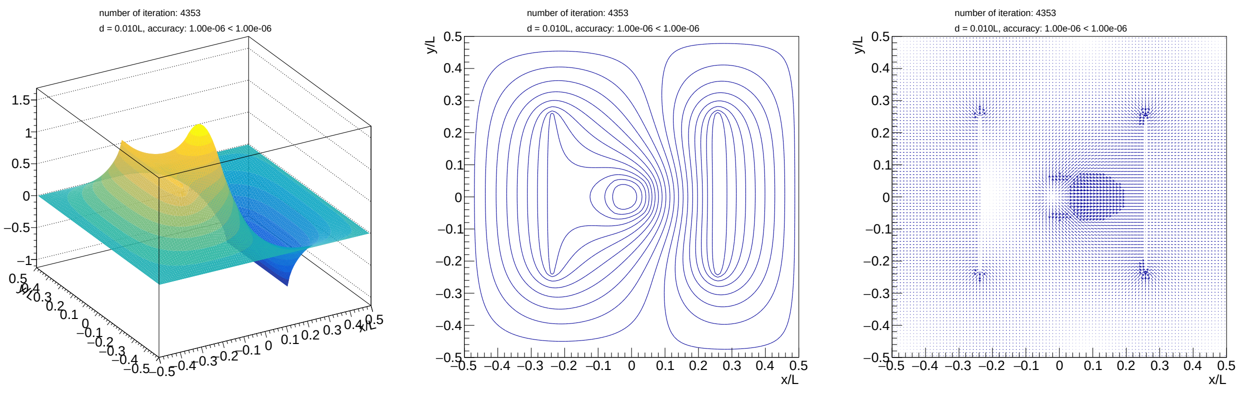

Figure 11: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 0.5L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.01L$. Set average error is $acc = 10^{-6}$. Boundary of the square is set to have $V = 0$.

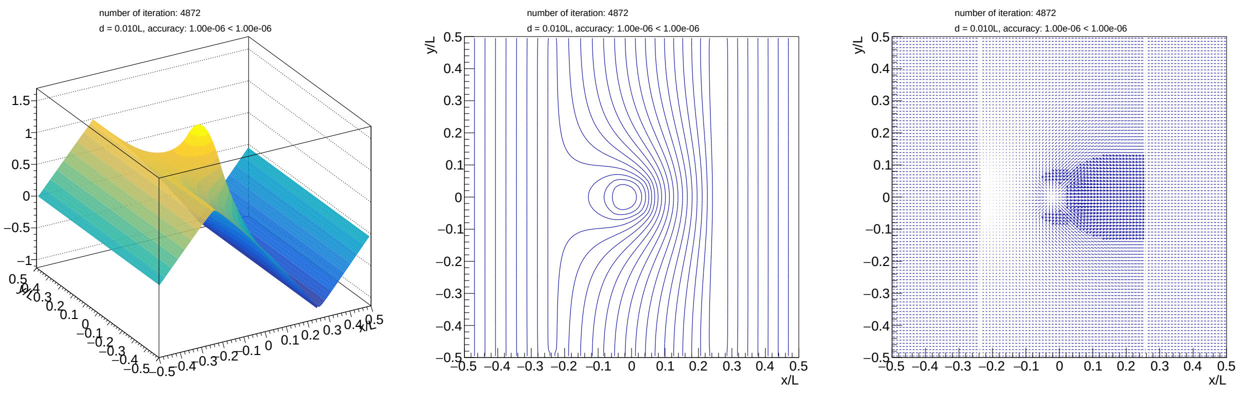

Figure 12: The potential mesh plot (left), potential contour plot (middle), and electric field plot (right) calculated by the Jacobi relaxation. The plate widths are $a = 1.0L$, and the distance between them is $b = 0.5L$. The grids are squares with edge $d = 0.01L$. Set average error is $acc = 10^{-6}$. Periodic boundary condition is applied.

Relevant code:

For the input charge density

```
//------------------------------------------------------------------------//

void Capacitor2D::set_rho(double x1, double x2, double y1, double y2, double r) {
        int i1 = int((x1+0.5)/_d+0.5);
        int i2 = int((x2+0.5)/_d+0.5);
        int ni = fabs(i2-i1) + 1;
        int j1 = int((y1+0.5)/_d+0.5);
        int j2 = int((y2+0.5)/_d+0.5);
        int nj = fabs(j2-j1) + 1;
        for(int j=j1; j<=j2; j++) {
                for(int i=i1; i<=i2; i++) {
                        _rho[j][i] = r/ni/nj;
                }
        }
}

//------------------------------------------------------------------------//
```

For the periodic case

```
//------------------------------------------------------------------------//

void Capacitor2D::cal_once_Jacobi_periodic() {

        if(!check()) return;

        vector< vector<double> > Vnew = _V;
        _tmp_acc = 0;

        for(int j=0; j<_N; j++) {
                for(int i=0; i<_N; i++) {
                        if(_V[i][j]==1 || _V[i][j]==-1) continue;
                        int jl = j-1; if(jl < 0) jl += _N;
                        int jr = j+1; if(jl>=_N) jr -= _N;
                        int il = i-1; if(il < 0) il += _N;
                        int ir = i+1; if(ir>=_N) ir -= _N;
                        Vnew[i][j] = (_V[il][j] + _V[ir][j] + _V[i][jl] + _V[i][jr])*0.25;
                        Vnew[i][j] += _rho[i][j];
                        _tmp_acc += fabs(Vnew[i][j] - _V[i][j]);
                }
        }

        _tmp_acc = _tmp_acc/_N/_N;

        _V = Vnew;

        _n_iter ++;
}

//------------------------------------------------------------------------//
```