PHYS580 Lab08 Report

Yicheng Feng
PUID: 0030193826

October 15, 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:

- `vector_field_2d.h` and `vector_field_2d.cxx` for the class `TVectorField2D` to plot the 2D vector field.
- `loop_magnetic_field.h` and `loop_magnetic_field.cxx` for the class `LoopMagneticField` to calculate the magnetic field of both the discrete loops and the helical coil.
- `lab8.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/lab8`.

(1) The two starter codes provided (`loop.m` and `loop_calculate_field.m`) use the simple rectangular panel method of integration to compute the magnetic field **B** of a current loop. The loop is in the x-y plane and is centered at the origin, just like in the setup discussed in class. (By the way, for this problem, the rectangular panel method is actually the same as the trapezoidal rule. Why?) Extend the programs (or create your own equivalent ones) to calculate the **B** field of two identically shaped parallel loops that share a common axis (the z-axis) and are situated symmetrically about the origin with their respective planes a distance d apart. First, calculate the field when the loops carry equal currents in the same direction, which is the Helmholtz coil configuration that is known to produce a nearly uniform magnetic field at the center. Then, investigate what happens if equal currents are carried in the opposite directions.

Physics explanation:

For this problem, the rectangular panel method is actually the same as the trapezoidal rule. Why?

This is because that the integration is over a loop, $f(0) = f(2\pi)$, where $f(\theta)$ is the function to be integrated.

$$\text{rectangular panel: } \int_0^{2\pi} f(\theta)\mathrm{d}\theta \approx \sum_{k=0}^{N-1} f\left(\frac{2\pi k}{N}\right)\frac{2\pi}{N} \tag{1}$$

$$\begin{aligned}\text{trapezoidal panel: } \int_0^{2\pi} f(\theta)\mathrm{d}\theta \approx & f(0)\frac{2\pi}{2N} + \sum_{k=1}^{N-1} f\left(\frac{2\pi k}{N}\right)\frac{2\pi}{N} + f(2\pi)\frac{2\pi}{2N} \\ = & f(0)\frac{2\pi}{N} + \sum_{k=1}^{N-1} f\left(\frac{2\pi k}{N}\right)\frac{2\pi}{N} = \sum_{k=0}^{N-1} f\left(\frac{2\pi k}{N}\right)\frac{2\pi}{N}\end{aligned} \tag{2}$$

From the equations above, we can see the two methods are the same in this problem.

For the geometry of the two loops, we put one centering at $(x, y, z) = (0, 0, 0.4)$ and the other at $(x, y, z) = (0, 0, -0.4)$, so $d = 0.8$. The loops are parallel to the x-y plane and have radius $R = 0.3$.

For the current, we set $\mu_0 I = 4\pi$ in numerical calculate.

First, we set the currents of the two loops in the same direction (Helmholtz coil). The results are shown in Fig. 1. In the left pad, we can see the magnetic field looks like constant near the origin. In the right pad, we can see the origin is at the bottom of the valley, so the first derivative of B_z is 0 in z direction.

Then, we set the currents of the two loops in the opposite directions. The results are shown in Fig. 2. In the left pad, we can see the magnetic field looks like zero near the origin. In the right pad, we can see from the symmetry $B_z = 0$ at the origin.

Plots:

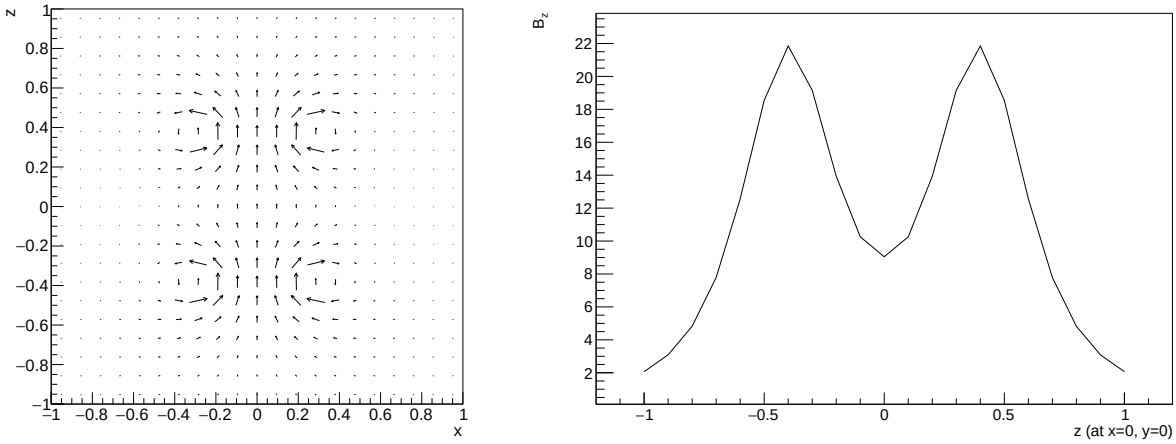

Figure 1: Two identical loops are centering at $(x, y, z) = (0, 0, 0.4)$ and $(0, 0, -0.4)$, with the same currents. The left pad shows the magnetic field (B_x and B_z) in the x-z plane with $y = 0$. The right pad shows the B_z magnitude depending on z at $x = y = 0$.

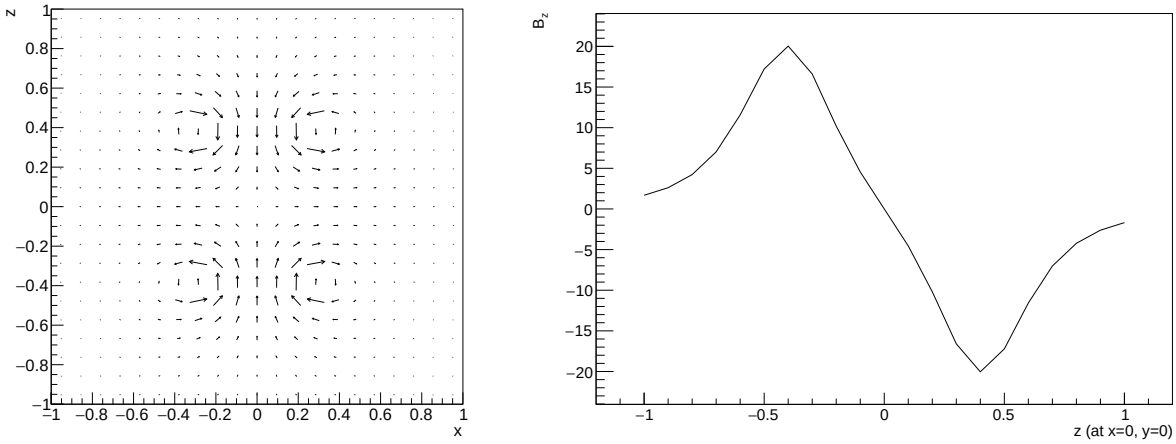

Figure 2: Two identical loops are centering at $(x, y, z) = (0, 0, 0.4)$ and $(0, 0, -0.4)$, with the opposite currents. The left pad shows the magnetic field (B_x and B_z) in the x-z plane with $y = 0$. The right pad shows the B_z magnitude depending on z at $x = y = 0$.

Relevant code:

For the initialization of the two loops configuration

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

LoopMagneticField::LoopMagneticField(int n_phi, double R, int n_loop, double pitch, double d) {

        _n_loop = n_loop;
        _n_phi = n_phi;
        _R = R;
        _pitch = pitch;
        _d = d;
        _mod = 0;

        _M = int(1.0/_d+0.5);
        _d = 1.0/_M;
        _N = 2*_M+1;

        for(int i=0; i<_n_loop; i++) {
                _z_loop.push_back(-0.5*_pitch*(_n_loop-1) + i*_pitch);
                _I_loop.push_back(1.0);
        }
}

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

For the rectangular panel integration for the Biot-Savart law

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

vector<double> LoopMagneticField::cal_one_grid(double x, double y, double z) {

        double bx = 0;
        double by = 0;
        double bz = 0;

        double dphi = 2*M_PI/_n_phi;

        for(int i_loop=0; i_loop<_n_loop; i_loop++) {
                for(int i=0; i<_n_phi; i++) {
                        double phi = i*dphi;
                        double dlx = -_R*dphi*sin(phi);
                        double dly =  _R*dphi*cos(phi);
                        double dlz = 0;
                        double rx = x - _R*cos(phi);
                        double ry = y - _R*sin(phi);
                        double rz = z - _z_loop[i_loop];
                        double r = sqrt(rx*rx + ry*ry + rz*rz);
                        if(r>_R*1e-4) {
                                bx += _I_loop[i_loop]*(dly*rz - dlz*ry)/(r*r*r);
                                by += _I_loop[i_loop]*(dlz*rx - dlx*rz)/(r*r*r);
                                bz += _I_loop[i_loop]*(dlx*ry - dly*rx)/(r*r*r);
                        }
                }
        }

        vector<double> b(3);
        b[0] = bx;
        b[1] = by;
        b[2] = bz;

```

```
34          return b;
35  }
36
37  //--------------------------------------------------------------------------//
38
39  void LoopMagneticField::cal() {
40
41          for(int k=0; k<_N; k++) {
42                  double z = _d*(k-_M);
43                  vector<double> tmpbx;
44                  vector<double> tmpby;
45                  vector<double> tmpbz;
46                  for(int i=0; i<_N; i++) {
47                          double x = _d*(i-_M);
48                          vector<double> b;
49                          if(_mod==0) b = cal_one_grid(x,0,z);
50                          if(_mod==1) b = cal_one_grid_helical(x,0,z);
51                          tmpbx.push_back(b[0]);
52                          tmpby.push_back(b[1]);
53                          tmpbz.push_back(b[2]);
54                  }
55                  _bx.push_back(tmpbx);
56                  _by.push_back(tmpby);
57                  _bz.push_back(tmpbz);
58          }
59  }
60
61  //--------------------------------------------------------------------------//
```

(2) Extend your programs further to implement the same calculation for a helical coil of multiple loops (i.e., a current-carrying wire wrapped around a cylinder). The coil is centered at the origin, its axis coincides with the z axis, and it has a given pitch P (i.e., the location of the wire advances by distance P in the axial direction as the angle of winding along the wire goes through 2π). Using your program, calculate the magnetic field both inside and outside the coil for various lengths and numbers of winding, with the current set such that $\mu_0 I = 4\pi$. Show results for at least one case with loose winding, illustrating how **B** leaks through the coil, and a case with tight winding, showing how **B** approaches the field of an ideal solenoid.

Physics explanation:

For the geometry of this problem, we set solenoid centering at the z-axis from $z = -0.5$ to $z = 0.5$, so $l = 1.0$. The radius of the circules is $R = 0.3$. The pitch is determined by the total number of loops (N), $P = l/N$.

For the **Loose winding** case, we set the number of loops $N = 3$, so the pitch $P = l/N = 0.3333$. The results are shown in Fig. 3. We can see that the magnetic field **B** leaks near the edges of the solenoid $x = \pm 0.3$.

For the **Tight winding** case, we set the number of loops $N = 30$, so the pitch $P = l/N = 0.0333$. The results are shown in Fig. 4. There is no visible **B** leak in the left plot. Moreover, the magnetic field seems constant along x-axis inside the solenoid.

As instructed by the NOTE, the analytical B_z along the z-axis for the solenoid from $z = -L/2$ to $z = L/2$ is

$$B_z = \frac{1}{2}\mu_0 n I \left(\frac{L/2 + z}{\sqrt{(L/2 + z)^2 + R^2}} + \frac{L/2 - z}{\sqrt{(L/2 - z)^2 + R^2}} \right). \tag{3}$$

We plug in numbers and get the analytical results shown as the red dashed lines in Fig. 3 and Fig. 4. The numerical and analytical results are very close to each other.

For both the loose winding and tight winding cases, the B_z distribution along the z-axis are close to the ideal solenoid.

Plots:

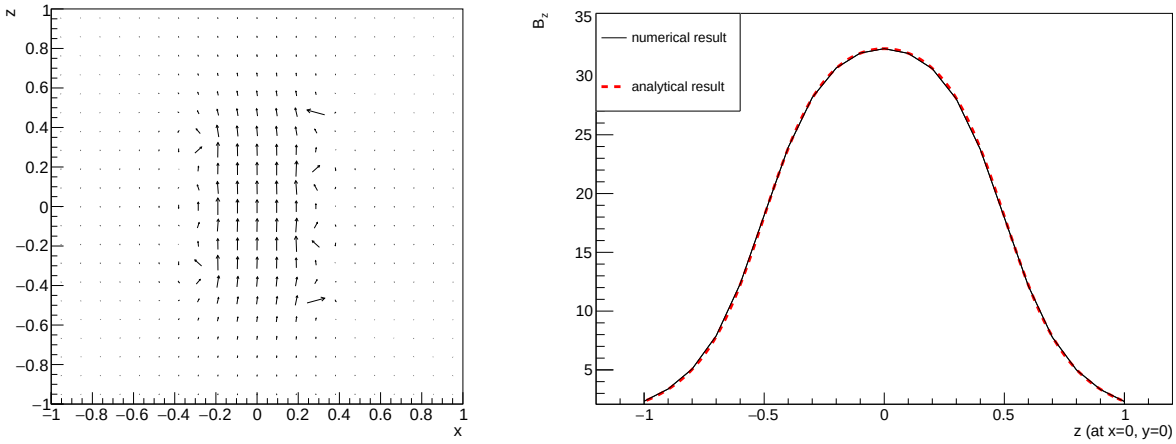

Figure 3: **Loose winding:** Helical coil are centering at the z-axis from $z = -0.5$ to $z = 0.5$ with winding number 3. The left pad shows the magnetic field (B_x and B_z) in the x-z plane with $y = 0$. The right pad shows the B_z magnitude depending on z at $x = y = 0$.

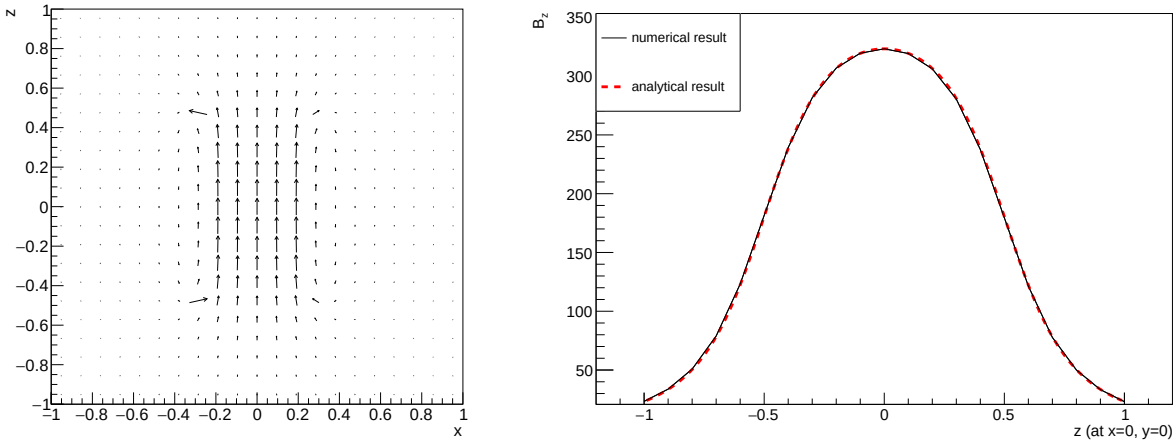

Figure 4: **Tight winding:** Helical coil are centering at the z-axis from $z = -0.5$ to $z = 0.5$ with winding number 30. The left pad shows the magnetic field (B_x and B_z) in the x-z plane with $y = 0$. The right pad shows the B_z magnitude depending on z at $x = y = 0$.

Relevant code:

For the rectangular panel integration for the Biot-Savart law

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

vector<double> LoopMagneticField::cal_one_grid_helical(double x, double y, double z) {

        double bx = 0;
        double by = 0;
        double bz = 0;

        double dphi = 2*M_PI/_n_phi*_n_loop;

        double z_low = -0.5*_pitch*_n_loop;

        for(int i=0; i<_n_phi; i++) {
                double phi = i*dphi;
                double dlx = -_R*dphi*sin(phi);
                double dly =  _R*dphi*cos(phi);
                double dlz =  _pitch/2.0/M_PI;
                double rx = x - _R*cos(phi);
                double ry = y - _R*sin(phi);
                double rz = z - (z_low+_pitch/2.0/M_PI*phi);
                double r = sqrt(rx*rx + ry*ry + rz*rz);
                if(r>_R*1e-4) {
                        bx += _I_loop[0]*(dly*rz - dlz*ry)/(r*r*r);
                        by += _I_loop[0]*(dlz*rx - dlx*rz)/(r*r*r);
                        bz += _I_loop[0]*(dlx*ry - dly*rx)/(r*r*r);
                }
        }

        vector<double> b(3);
        b[0] = bx;
        b[1] = by;
        b[2] = bz;

        return b;
}

//------------------------------------------------------------------------//

void LoopMagneticField::cal() {

        for(int k=0; k<_N; k++) {
                double z = _d*(k-_M);
                vector<double> tmpbx;
                vector<double> tmpby;
                vector<double> tmpbz;
                for(int i=0; i<_N; i++) {
                        double x = _d*(i-_M);
                        vector<double> b;
                        if(_mod==0) b = cal_one_grid(x,0,z);
                        if(_mod==1) b = cal_one_grid_helical(x,0,z);
                        tmpbx.push_back(b[0]);
                        tmpby.push_back(b[1]);
                        tmpbz.push_back(b[2]);
                }
                _bx.push_back(tmpbx);
                _by.push_back(tmpby);
                _bz.push_back(tmpbz);
```

```
58              }
59  }
60
61  //-------------------------------------------------------------------------//
```