Modeling the Earths Magnetic Field Interaction with the Solar Wind: Kernel Review

Justin R Bergonio

24 April 2012

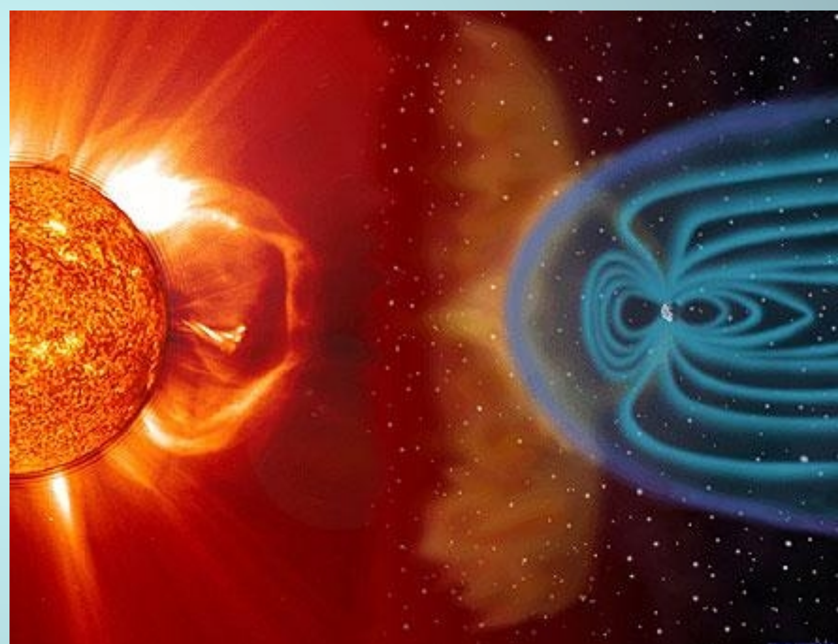
PHYS 305/Professor Varner

Table of Contents

- Introduction
- Main Problem
- Possible Solution(s)
- Kernel
- Lessons Learned
- Status (#Progress)

Introduction

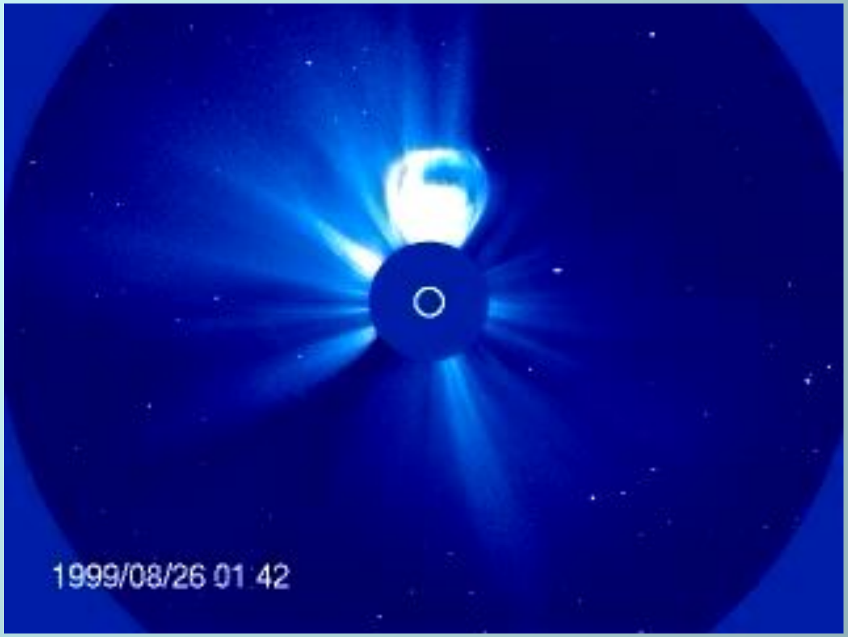
- Earth's magnetic field
  - Approx. dipole field
  - Deflects charged particles
  - $2.5 - 6.5 \times 10^{-5}$ T
- Solar wind
- Solar flares
- Coronal Mass Ejections (CME's)



Introduction

Solar Flare

CME



Main Problem

- Modeling the Earth's magnetic field:
  - In simple magnetic dipole case
  - Against solar wind
  - Interaction with solar flares and CME's

Possible Solutions

- The Random
  - Generate random points (x,y,z)
  - Using gnuplot, obtain contours
- The Loops
  - Use three for-loops (generates an x, y, z)
  - Using gnuplot, obtain contours
  - Plot field lines (2-D/3-D)
- Equations: $B_r = -2B_0\left(\frac{R_E}{r}\right)^3 \cos\theta$ $\quad B_\theta = -B_0\left(\frac{R_E}{r}\right)^3 \sin\theta$ $\quad |B| = B_0\left(\frac{R_E}{r}\right)^3 \sqrt{1+\cos^2\theta}$

Kernel: The Random

```
ofstream myfile ("FP_MagR.txt");
main(){
    int Number = 0;
    float B0, Br, Baz, Btot, mag_perm, m, radius, rho, R_Earth;
    int N_Total = 12940, step_size = 100;
    float initial;

    initial = -(R_Earth + 100.0);
    B0 = 3.12*pow(10.0,-5);                          // mean-value magnetic field (T)
    R_Earth = 6370.0;                                // radius of Earth in meters (km)

    srand( (unsigned) time (NULL));

    for (int i = 1; i < N_Total; ++i){
        float x[N_Total], y[N_Total], z[N_Total];
        float floor = 0.0 - (R_Earth + pow(10.0,6)), ceiling = R_Earth + pow(10.0,6), range = (ceiling - floor);

        x[i] = floor + ((range * rand()) / (RAND_MAX + 1.0));
        y[i] = floor + ((range * rand()) / (RAND_MAX + 1.0));
        z[i] = floor + ((range * rand()) / (RAND_MAX + 1.0));

        radius = sqrt(pow(x[i],2) + pow(y[i],2));               // radius from center of Earth 2D
        rho = sqrt(pow(z[i],2) + pow(radius,2));             // radius from center of Earth 3D

        Br = -2.0*B0*pow( (R_Earth/radius),3 )*(z[i]/rho);
        Baz = -B0*pow( (R_Earth/radius),3 )*(radius/rho);
        Btot = Br + Baz;
        //Btot = -B0*pow( (R_Earth/radius),3 )*sqrt( 1.00 + 3*pow((z[i]/rho),2));


        cout.precision(1);
        cout.setf(ios::fixed);
        cout << x[i] << "\t" << z[i] << "\t";
        cout.precision(15);
        cout.setf(ios::fixed);
        cout << Btot << endl;

        myfile.precision(1);
        myfile.setf(ios::fixed);
        myfile << x[i] << "\t" << z[i] << "\t";
        myfile.precision(15);
        myfile.setf(ios::fixed);
        myfile << Btot << endl;

        //cout << x << "\t" << y << "\t" << z << "\t" << Br << "\t" << Baz << "\t" << Btot << endl;
        //myfile << x << "\t" << y << "\t" << z << "\t" << Br << "\t" << Baz << "\t" << Btot << endl;
    }
    myfile.close();
```

Kernel: The Loops

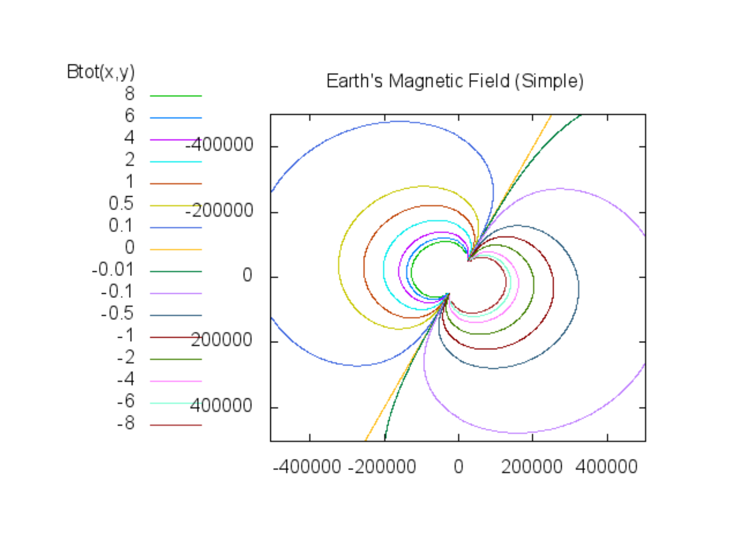
```
main(){
    int N_Total;
    double B0, Br, Baz, Btot, Btot2, rho, R_Earth, x, y, z, radius;
    double initial, step;

    B0 = 3.12*pow(10.0,-5);                          // mean-value magnetic field (T)
    R_Earth = 6370.0;                                 // radius of Earth in meters (km)
    initial = -(R_Earth + 1000.0);
    step = 1.0;
    N_Total = 10000;

    //srand( (unsigned) time (NULL));

    /*for(int i = 0; i <= N_Total; ++i ){
        double x[N_Total], y[N_Total], z[N_Total];

            x[i] = initial + i*step;*/

                for(int n = 0; n <= N_Total; ++n ){
                    double x, y[N_Total], z[N_Total];

                    x = 0.0;

                    y[n] = initial + n*step;

                        for(int m = 0; m <= N_Total; ++m ){

                            z[m] = initial + m*step;

                            radius = sqrt( pow(x,2) + pow(y[n],2) );
                            rho = sqrt( pow(radius,2) + pow(z[m],2) );

                            Br = -2.0*B0*pow( (R_Earth/rho),3 )*(z[m]/rho);
                            Baz = -B0*pow( (R_Earth/rho),3 )*(radius/rho);
                            Btot =  B0*pow( (R_Earth/rho),3 )*sqrt(1.0 + 3*pow((z[m]/rho),2) );
                            //Btot2 = sqrt( pow(Br,2) + pow(Baz,2) );

                            if( Btot > (4.99*pow(10.0,-5)) && Btot < (5.01*pow(10.0,-5)) ){
                                cout.precision(1);
                                cout.setf(ios::fixed);
                                cout /*<< x[i] << "\t"*/ << y[n] << "\t" << z[m] << "\t" << rho << endl;
                                //cout.precision(6);
                                //cout.setf(ios::fixed);
                                //cout << Btot << "\t" << Btot2 << endl;

                                myfile.precision(1);
                                myfile.setf(ios::fixed);
                                myfile /*<< 00000.0 << "\t"*/ << y[n] << "\t" << z[m] << endl;
                                //myfile.precision(8);
                                //myfile.setf(ios::fixed);
                                //myfile << Btot << endl;
                            }
                        }
                    //}
            }
```

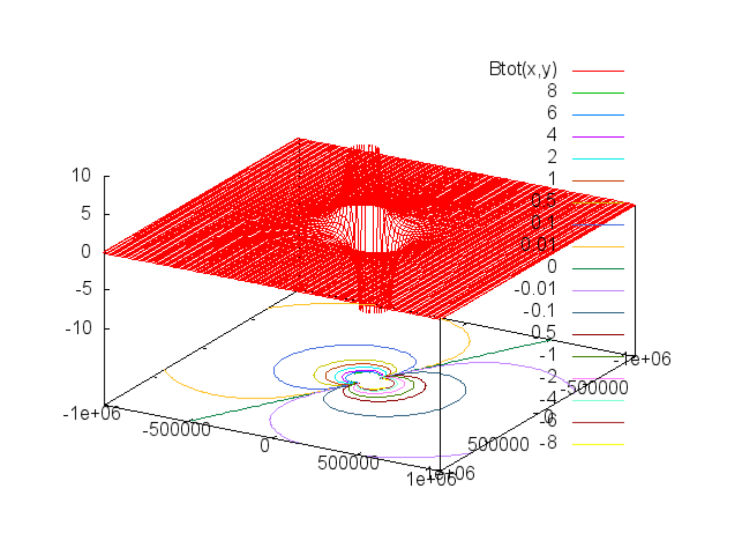
Lessons Learned

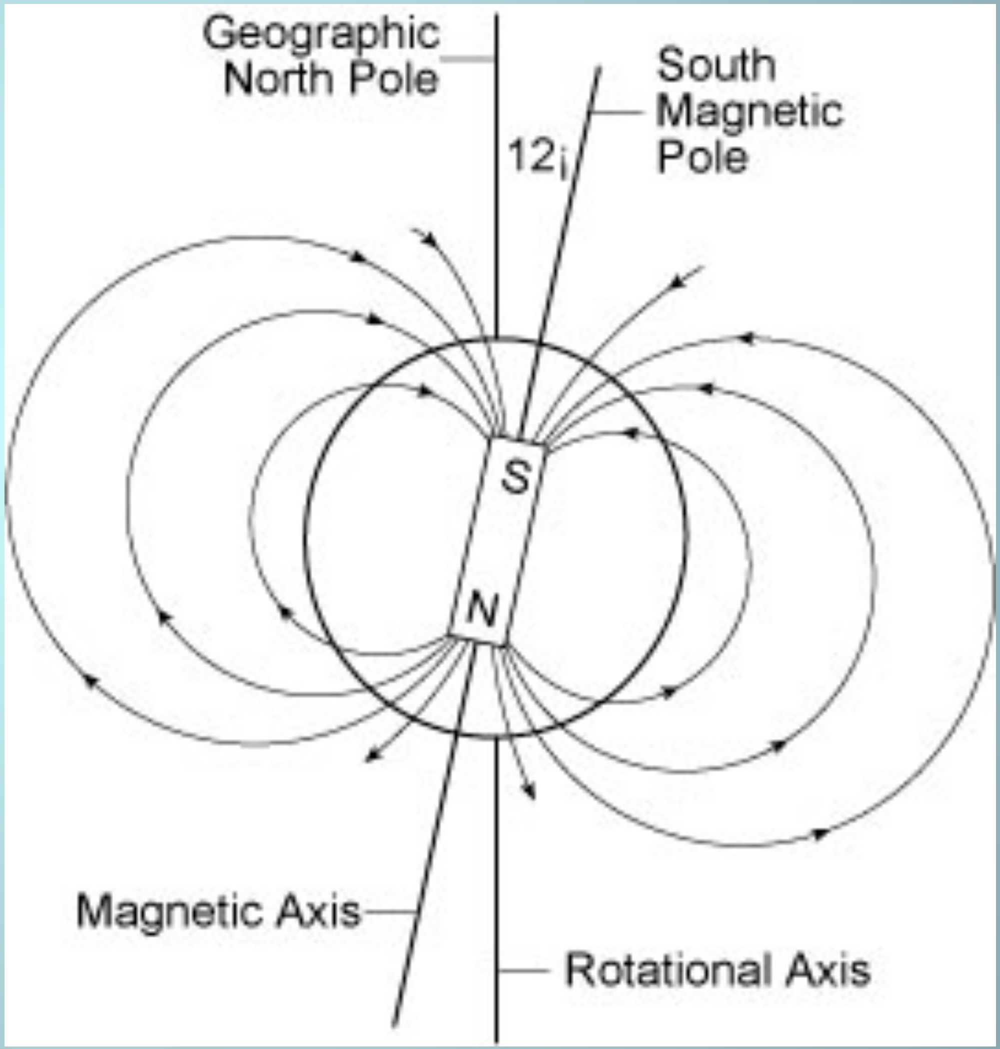
"I have not failed 10,000 times.
I have successfully found 10,000 ways
that will not work."
-Thomas Edison

Lessons Learned

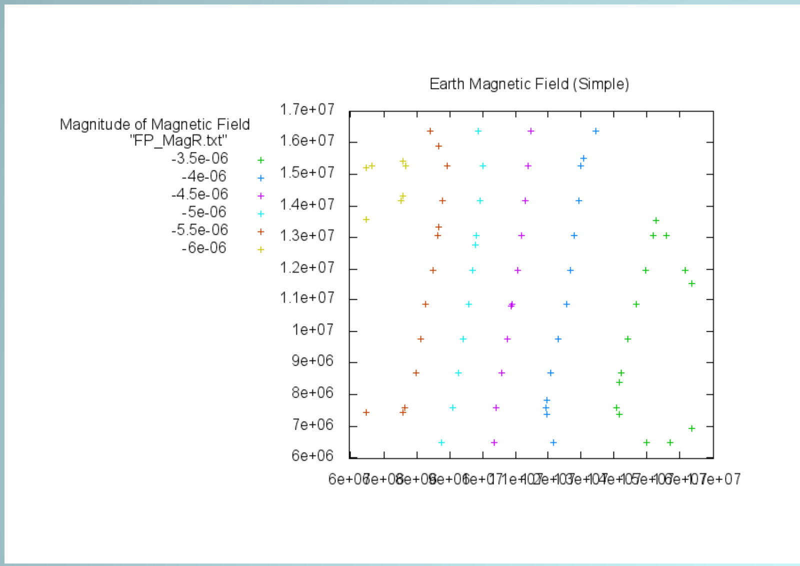
Status: Gnuplot

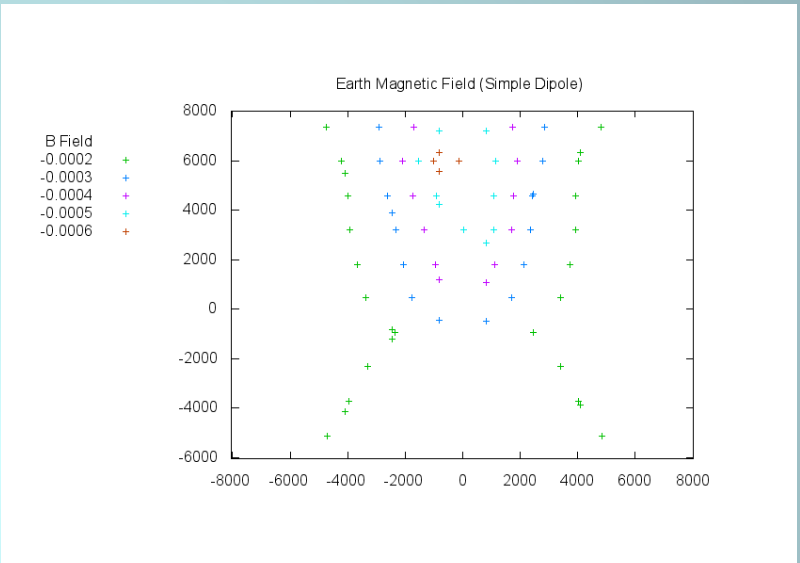


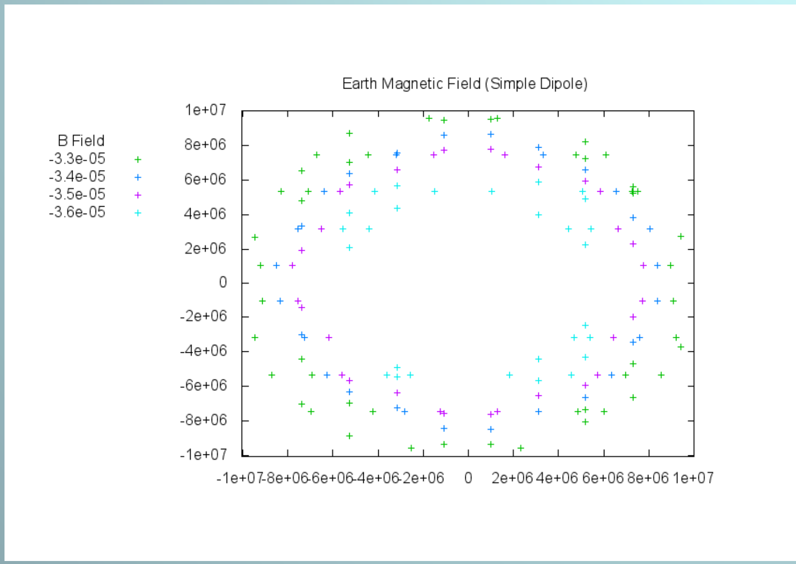


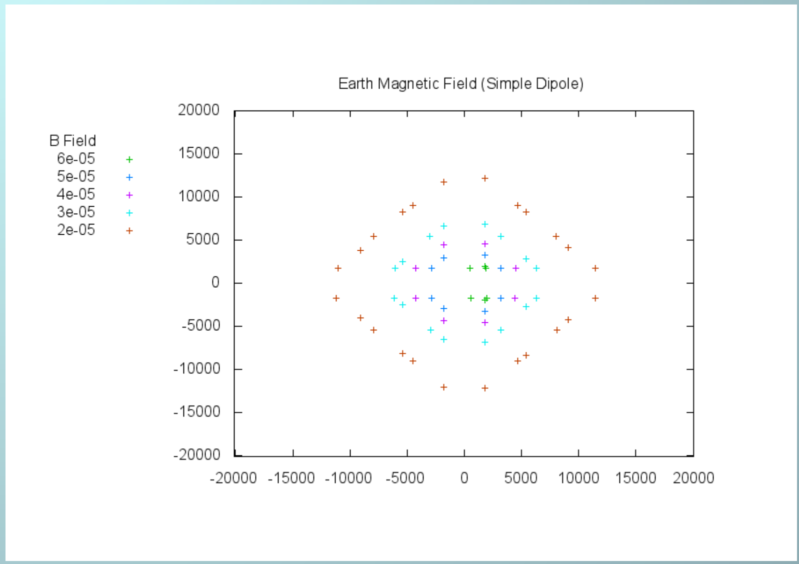


Status: C++ (The Random)

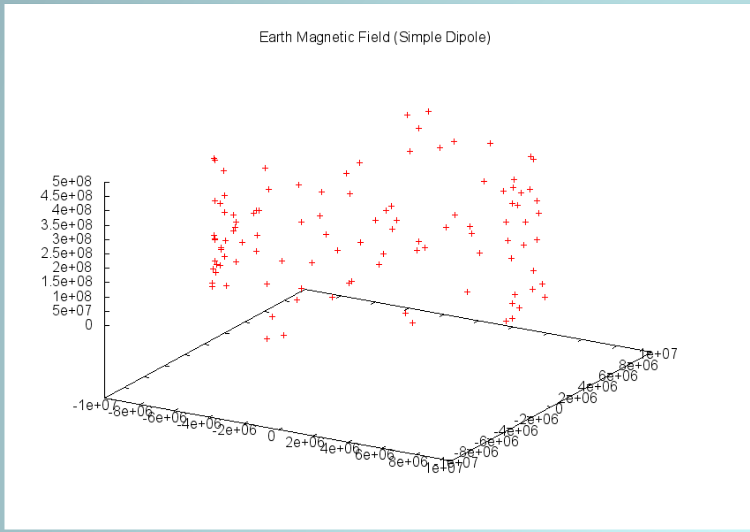


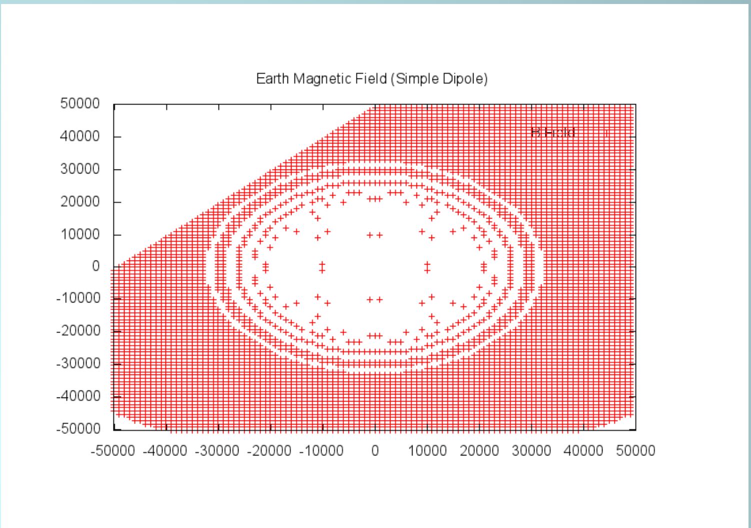


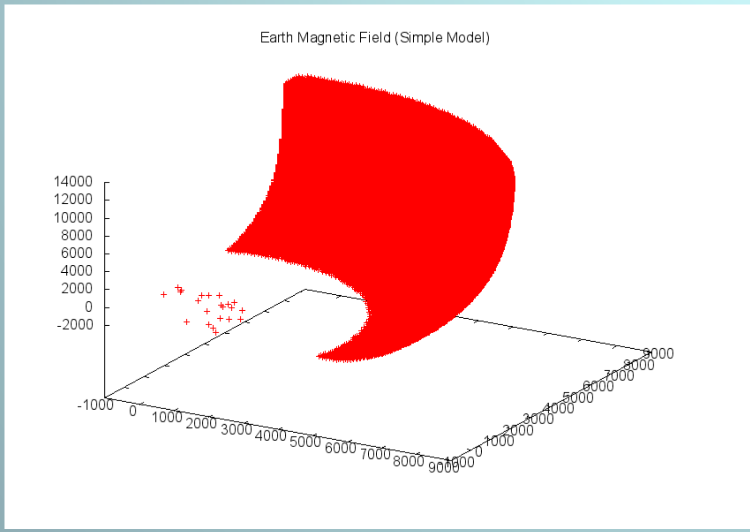


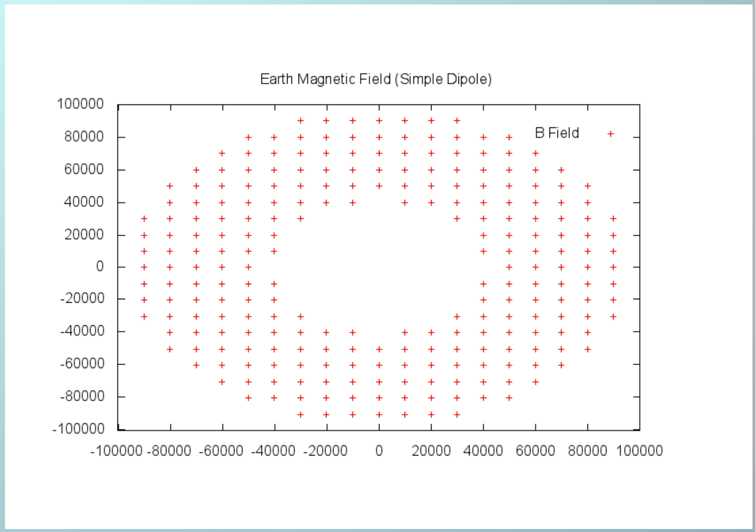


Status: C++ (The Loops)

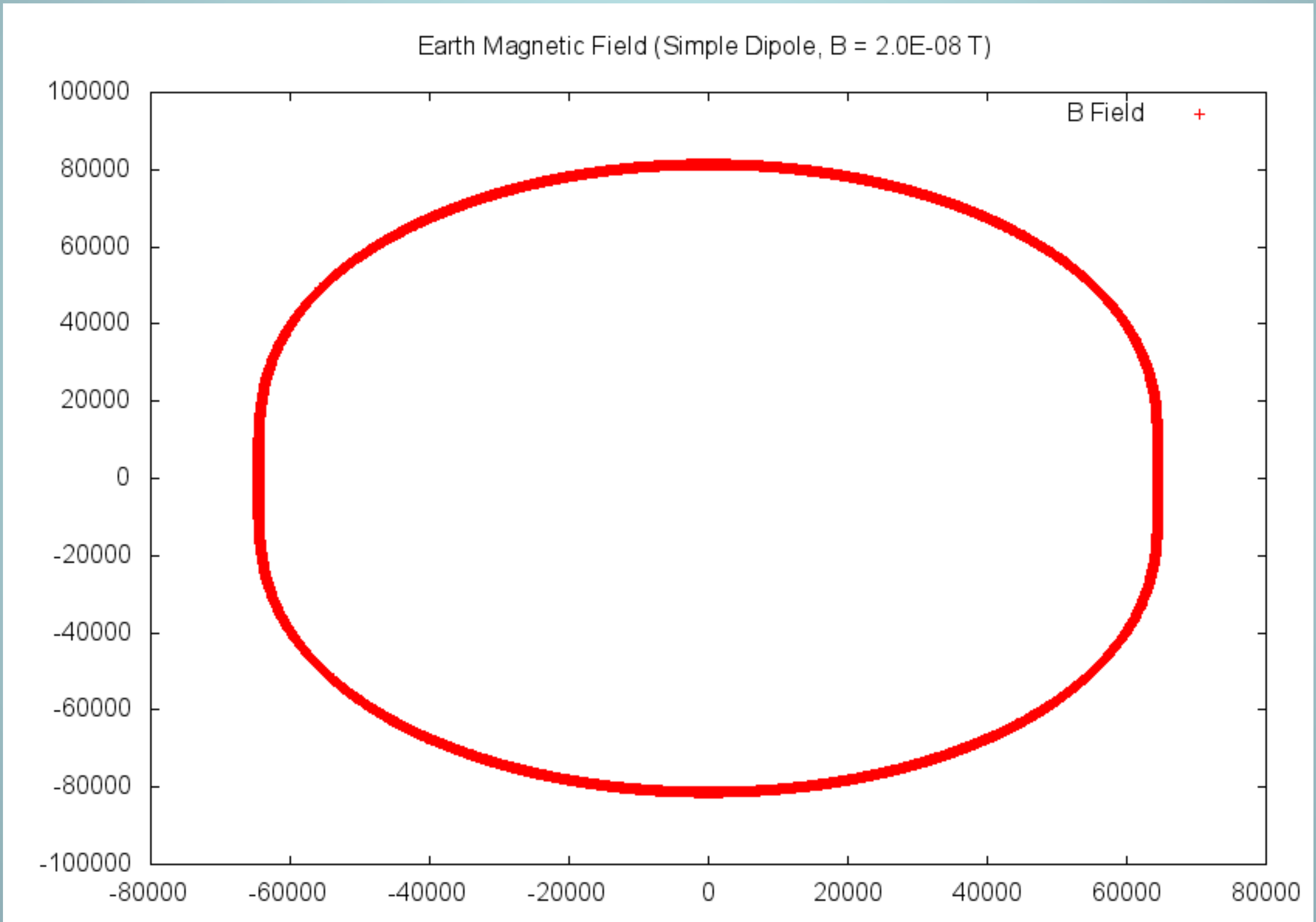








Status: WIN (almost)



Conclusion

- Remember spherical coordinates
- Be more effective
  - “Tell me what you want, what you really, really want”
- Be extra patient

And if you have an idea…

Just Go With It