REFERENCES

[1] J. R. Conrad, 1988. Method and apparatus for plasma source ion implantation, United States Patent 4,764,394.

[2] F. F. Chen., 1984. Introduction to Plasma Physics and controlled fusion. 2nd ed. USA: Plenum Press,

[3] E. Knystautas, 2005. Engineering Thin Films and Nanostructures with Ion Beams. USA: Taylor & Francis,

[4] J. R. Conrad, 1989. Plasma Source Ion Implantation: A New Approach to Ion Beam Modification of Materials, Materials Science and Engineering All 6: 197-203.

[5] C. Liu, 2003. Ion dynamics of pulsed plasma source ion implantation in the sheath of a hemispherical bowl-shaped target, Surface and Coatings Technology 171: 119–123.

[6] X. B. Tiana, 2004. Two-dimensional numerical simulation of non-uniform plasma immersion ion implantation, Surface & Coatings Technology 186: 47– 52.

[7] B. Rauschenbach, 2003. Plasma-sheath expansion around trenches in plasma immersion ion implantation, Nuclear Instruments and Methods in Physics Research.B 206: 803–807.

[8] B. Briehl, 2002. Simulation of sheath and presheath dynamics in PIII, Surface and Coatings Technology 156: 131–135.

[9] S. Mandl, 1997. Measured and calculated dose distribution for 2D plasma immersion ion implantation, Surface and Coatings Technology 93: 229-233.

[10] K. C. Walter, 1998. Advances in PSII techniques for surface modification, Surface and Coatings Technology 103–104: 205–211.

[11] H. Erramli, et al., 2007. A Monte Carlo computer code for evaluating energy loss of 10 keV to 10 MeV ions in amorphous silicon materials. Nucl. Instr. and Meth. in Phys. Res.,B 263: 127–131.

[12] J. P. Biersack, L. Haggmark, 1980. Nucl. Instr. and Meth. 174: 257.

[13] J. F. Ziegler, et al., 1985. The stopping and range of ions in solids. New York: Pergamon,

[14] M. H. Shapiro, P. Lu, 2004. The influence of the ion–atom potential on molecular dynamics simulations of sputtering, Nucl. Instr. and Meth. in Phys. Res., B 215: 326–336.

[15] S. Zhang, et. al., 2004. Experimental and molecular dynamics simulation studies of friction behavior of hydrogenated carbon films, Surface and Coatings Technology 177 –178: 818–823.

[16] C. – J. Chu, T. – C. Chen, 2006. Surface properties of film deposition using molecular dynamics simulation, Surface and Coatings Technology 201: 1796-1804

[17] R. J. Arsenault, et al, 1986. Computer Simulationtion in Material Science, USA: ASM International.

[18] A. Stasikowski, 2007. The structural properties of alkali metal atoms doped noble gas clusters, Radiation Physics and Chemistry 76: 607–611.

[19] J. Merimaa, et. al., 2000. An interactive simulation program for visualizing complex phenomena in solids, Computer Physics Communications 124: 60–75.

[20] Z. Tao, et. al., 1998. Stopping power for MeV ^{12}C ions in solids, Nucl. Instr. and Meth. in Phys. Res., B 135: 169-174.

[21] D. R. Olander, 1985. Fundamental Aspects of Nuclear Reactor Fuel Element, Califonia: Technical Information Center Energy Research and Development Adminiistration,

[22] G. S. Was, 2007. Fundamental of radiation material Science, Berlin, Heidelberg: S pringer-Verlag,

[23] J. F. Zigler, 2006. Program SRIM/TRIM. Version 2006. Available from: http://www.srim.org

[24] C. K. Birdsall, A. B. Langdon, 1985 Plasma Physics via computer simulation 1st ed. Singapore: McGrow-Hill,

[25] K. A. HoffMann, S. T. Chiang, 2000. Computational Fliud Dynamic 4th ed. USA: Wichita,

APPENDICES

APPENDIX A

The Input files and codes for plasma source of ion implantation

PARTICAL DENSITY:

1.00e2

LengthX:

0.10000

LengthY:

0.1000

LengthZ:

0.1000

INTERVAL TIME:

1.00e-8

TIME STEP:

1

BIAS POTENTIAL:

-2000.00

TEMPERATURE:

232000.00

----------Input file----------

```
#define q          1.600e-19
#define MI         1.670e-27
#define ME         9.109e-31
#define f          8.850e-12
#define Kb         1.380e-23
#define PartX      20
#define PartY      20
#define PartZ      20
#define tolerate   1.00
#define Particle   100000
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

long TIME_STEP ;

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP;

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

long step;

void main()

{


F_Input();

srand(3);

RandomIX();

RandomIY();

RandomIZ();


//////////////////////////////////////////////
for (step=0 ; step < TIME_STEP ; step++)

{

    F_Basement();

    F_PrintMove();

    F_IDensity(SIX,SIY,SIZ);

    F_PrintNormal2();

    //F_PrintPHI2();

    F_Phi();

    F_Electric();

    F_Magnetic();

    Runge_Kutta();

    //F_PrintParticle();

    //F_PrintPHI();

    //F_PrintPHI2();

    F_PrintNormal();

    ClearPhi();
```

```
    printf("%i\n",step);

}

////////////////////////////////////////////////
return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

long TIME_STEP ;

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP;

double DE[PartX][PartY][PartZ];

double dx,dy,dz;

double debye,Pdebye,Density ;

void F_Input()

{

FILE *finput,*foutput;

long Pipi,Pipj,Pipk;

finput = fopen("InPut.txt","r+t");

if(finput == NULL)

{
    printf("ERRorrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr !\n");

    exit(1);

}

else

{

    fseek(finput,17L, SEEK_CUR);
    fscanf(finput,"%lg\n",&Rho);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthX);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthY);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthZ);

    fseek(finput,14L, SEEK_CUR);
    fscanf(finput,"%lg\n",&t);

    fseek(finput,10L, SEEK_CUR);
    fscanf(finput,"%ld\n",&TIME_STEP);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&IPhi);

    fseek(finput,12L, SEEK_CUR);
    fscanf(finput,"%lg\n",&TEMP);


}

fclose(finput);
```

```
    dx = LengthX/PartX;
    dy = LengthY/PartY;
    dz = LengthZ/PartZ;
    Density = (Rho*Particle)/(LengthX*LengthY*LengthZ);
    debye =pow(((f*Kb*TEMP)/(Density*q*q)),0.5);
    Pdebye =Density*(4/3)*3.14*pow(debye,3);

foutput = fopen("c_data.txt","w+");

if(foutput == NULL)
{
    printf("ERRorrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr !\n");
    exit(1);
}
else
{
    fprintf(foutput,"Particles  %d \nRHO %e \n",Particle,Rho);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"LengthX %f \nLengthY %f \nLengthZ %f\n",LengthX,LengthY,LengthZ);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"PartX %d \nPartY %d \nPartZ %d\n",PartX,PartY,PartZ);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"dx %f \ndy %f \ndz %f\n",dx,dy,dz);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Interval Time %e \nTime STEP %d\n",t,TIME_STEP);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"IPHI %f \n",IPhi);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"tolerate %f \n",tolerate);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Temperature %f \n",TEMP);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Ion Density %e \n",Density);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Debye Length %f \n",debye);
```

```
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Particles in Debye Sphere %e \n",Pdebye);
    fprintf(foutput,"----------------------------\n");
}
fclose(foutput);
for (Pipi=0 ; Pipi < PartX ; Pipi++)
    {
        for (Pipj=0 ; Pipj < PartY ; Pipj++)
        {
            for (Pipk=0 ; Pipk < PartZ ; Pipk++)
            {
                DE[Pipi][Pipj][Pipk]= Density;
            }
        }
    }

return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle];

double LengthX;

void RandomIX()

{

long nix ;

for ( nix = 0 ; nix < Particle ; nix++)

    {

    do
        {

            SIX[nix] = LengthX*(1.00*rand()/RAND_MAX);

        }

    while((SIX[nix]>=LengthX)||(SIX[nix]<0.0));

    }

}

//RANDOM
```

RandomIX.c

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIY[Particle];

double LengthY;

void RandomIY()

{

long niy ;

for ( niy = 0 ; niy < Particle ; niy++)

    {

    do
        {

            SIY[niy] = LengthY*(1.00*rand()/RAND_MAX);

        }

    while((SIY[niy]>=LengthY)||(SIY[niy]<0.0));

    }


}
```

RandomIY.c

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIZ[Particle];

double LengthZ;

void RandomIZ()

{

long niz ;

for ( niz = 0 ; niz < Particle ; niz++)

    {

    do
        {

            SIZ[niz] =LengthZ*(1.00*rand()/RAND_MAX);

        }

    while((SIZ[niz]>=LengthZ)||(SIZ[niz]<0.0));

    }


}
```

RandomIZ.c

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double LengthX,LengthY,LengthZ,dx,dy,dz;

long base,step;

void F_Basement()

{

    for(base = 0; base < Particle; base ++)

    {

                if( (SIX[base] >= 7.00*dx) && (SIX[base] < 13.00*dx) &&
                    (SIY[base] >= 7.00*dy) && (SIY[base] < 13.00*dy) &&
                    (SIZ[base] >= 7.00*dz) && (SIZ[base] < 13.00*dz) )

                {

                    XXI1[base] = 0.00;

                    YYI1[base] = 0.00;

                    ZZI1[base] = 0.00;

                    do

                    {

                        SIX[base] = LengthX*(1.00*rand()/RAND_MAX);

                        SIY[base] = LengthY*(1.00*rand()/RAND_MAX);

                        SIZ[base] = LengthZ*(1.00*rand()/RAND_MAX);

                    }while( (SIX[base] >= 7.00*dx) && (SIX[base] < 13.00*dx)&&
                    (SIY[base] >= 7.00*dy) && (SIY[base] < 13.00*dy) &&
                    (SIZ[base] >= 7.00*dz) && (SIZ[base] < 13.00*dz) );


                }


    }

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

int  decimal, sign;

void F_PrintMove()

{

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

FILE *fMovement;


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

{
    fMovement=fopen("Movement.csv","a+t");

    if(fMovement == NULL)

    {

        printf("ERRorrrr-Movement.csv !\n");

        exit(1);

    }


else

{

        if(step == 0)
        {
        fprintf(fMovement,"step,");
        fprintf(fMovement,"SIX[2667],SIY[2667],SIZ[2667],XXI1[2667],YYI1[2667],ZZI1[266
7],");
```

```
        fprintf(fMovement,"SIX[25000],SIY[25000],SIZ[25000],XXI1[25000],YYI1[25000],ZZI
1[25000],");
        fprintf(fMovement,"SIX[50000],SIY[50000],SIZ[50000],XXI1[50000],YYI1[50000],ZZI
1[50000],");
        fprintf(fMovement,"SIX[75000],SIY[75000],SIZ[75000],XXI1[75000],YYI1[75000],ZZI
1[75000],");
        fprintf(fMovement,"SIX[7954] ,SIY[7954] ,SIZ[7954] ,XXI1[7954] ,YYI1[7954] ,ZZI
1[7954],");
        fprintf(fMovement,"SIX[64995] ,SIY[64995] ,SIZ[64995] ,XXI1[64995] ,YYI1[64995]
 ,ZZI1[64995],");
        fprintf(fMovement,"SIX[26613] ,SIY[26613] ,SIZ[26613] ,XXI1[26613] ,YYI1[26613]
 ,ZZI1[26613],");
        fprintf(fMovement,"SIX[28141] ,SIY[28141] ,SIZ[28141] ,XXI1[28141] ,YYI1[28141]
 ,ZZI1[28141],");
        fprintf(fMovement,"SIX[82996],SIY[82996],SIZ[82996],XXI1[82996],YYI1[82996],ZZI
1[82996],");
        fprintf(fMovement,"SIX[99999],SIY[99999],SIZ[99999],XXI1[99999],YYI1[99999],ZZI
1[99999]\n");
        }

fprintf(fMovement,"%i,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e
,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e,%e
,%e,%e,%e,%e,%e,%e,%e,%e,%e\n"
        ,step,
        SIX[2667],SIY[2667],SIZ[2667],XXI1[2667],YYI1[2667],ZZI1[2667],
        SIX[25000],SIY[25000],SIZ[25000],XXI1[25000],YYI1[25000],ZZI1[25000],
        SIX[50000],SIY[50000],SIZ[50000],XXI1[50000],YYI1[50000],ZZI1[50000],
        SIX[75000],SIY[75000],SIZ[75000],XXI1[75000],YYI1[75000],ZZI1[75000],
        SIX[7954] ,SIY[7954] ,SIZ[7954] ,XXI1[7954] ,YYI1[7954] ,ZZI1[7954],
        SIX[64995] ,SIY[64995] ,SIZ[64995] ,XXI1[64995] ,YYI1[64995] ,ZZI1[64995],
        SIX[26613] ,SIY[26613] ,SIZ[26613] ,XXI1[26613] ,YYI1[26613] ,ZZI1[26613],
        SIX[28141] ,SIY[28141] ,SIZ[28141] ,XXI1[28141] ,YYI1[28141] ,ZZI1[28141],
        SIX[82996],SIY[82996],SIZ[82996],XXI1[82996],YYI1[82996],ZZI1[82996],
        SIX[99999],SIY[99999],SIZ[99999],XXI1[99999],YYI1[99999],ZZI1[99999]);


}

fclose(fMovement);

}

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double DI[PartX][PartY][PartZ];

double dx,dy,dz,Rho,Time_Step;

void F_IDensity(double *IX,double *IY, double *IZ)
//void F_IDensity(double IX[Particle],double IY[Particle], double IZ[Particle])
{

double IN[PartX][PartY][PartZ];

long ILeft,IRight,IFront,IBack,ITop,IDown;

long ii,ij,ik ;

long ni ;


for (ii=0 ; ii < PartX ; ii++)

    {

    for (ij=0 ; ij < PartY ; ij++)

        {

        for (ik=0 ; ik < PartZ ; ik++)

            {

                IN[ii][ij][ik] = 0.00 ;

            }

        }
    }


for(ni=0; ni < Particle; ni++)

        {

            ILeft=(long)floor(IX[ni]/dx);

            if ( ILeft == PartX)

            {

                ILeft= 0;

            }


            IRight=ILeft+1;


            IBack = (long)floor(IY[ni]/dy);
```

```
            if ( IBack == PartY)
            {
                IBack= 0;
            }
            IFront = IBack+1;
            if ( IDown == PartZ)
            {
                IDown= 0;
            }
            IDown = (long)floor(IZ[ni]/dz);
            ITop = IDown+1;
            if((IX[ni]>=ILeft*dx)&&(IX[ni]<IRight*dx)
                &&(IY[ni]>=IBack*dy)&&(IY[ni]<IFront*dy)
                &&(IZ[ni]>=IDown*dz)&&(IZ[ni]<ITop*dz))
                    {
                        IN[ILeft][IBack][IDown]=IN[ILeft][IBack][IDown]+1;
                    }
            }
for (ii=0 ; ii < PartX ; ii++)
    {
    for (ij=0 ; ij < PartY ; ij++)
        {
        for (ik=0 ; ik < PartZ ; ik++)
            {
                DI[ii][ij][ik]= Rho*IN[ii][ij][ik]/(dx*dy*dz);
            }
        }
    }

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

long AOH ;

void F_PrintNormal2()

{

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

FILE *floveAOH2;


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

{
    floveAOH2=fopen("LoveAoh222.csv","a+t");

if(floveAOH2 == NULL)
    {

        printf("ERRorrrr-Love AOH2222222222222222.csv !\n");

        exit(1);

    }

else

{

    for( AOH=0 ; AOH < PartX ; AOH++)

        {

            fprintf(floveAOH2,"%e,",DI[AOH][9][9],DE[AOH][9][9]);
```

```
        }
        fprintf(floveAOH2,"\n");
}
fclose(floveAOH2);
}
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

char *B;

int  decimal, sign;

void F_PrintPHI2()

{

long p;

long pi,pj,pk;

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

FILE *fPhi_9;

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

if( step%100 ==0)

{
    B = _fcvt((float)step+1.00,0,&decimal,&sign);

    strcat(B,"_9.csv");

    fPhi_9 = fopen(B,"w+t");

    if(fPhi_9 == NULL)

    {

        printf("ERRorrrr-Phi record 9.csv !\n");

        exit(1);

    }

 else
```

```
{
for(pi = 0; pi < PartX; pi++)
    {
        for(pj =0 ; pj < PartY; pj++)
        {
            fprintf(fPhi_9,"%i,%i,%e,%e,%e\n",pi,pj,Phi[pi][pj][9],DI[pi][pj][9],DE[pi]
[pj][9]);
        }
    }
fclose(fPhi_9);
}
}
//-------------------------------------------------------------------------------------/
/
//-------------------------------------------------------------------------------------/
/


}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double Phi[PartX][PartY][PartZ];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double dx,IPhi,TIPhi,TEMP,Density ;

long step;

long TIME_STEP ;

void F_Phi()

{

long ii,jj,kk ;

long ipi,ini,jpj,jnj,kpk,knk;

double T_Phi[PartX][PartY][PartZ];

double Error[PartX][PartY][PartZ];

long Er[PartX][PartY][PartZ],Err;

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

    for (ii=0 ; ii < PartX ; ii++)

    {

        for (jj=0 ; jj < PartY ; jj++)

        {

            for (kk=0 ; kk < PartZ ; kk++)

            {

                T_Phi[ii][jj][kk] = 0.00 ;

                Phi[ii][jj][kk]   = 0.00 ;

                if( ii == 9 && jj == 9 && kk ==9)

                {

                    Phi[9][9][9] = IPhi ;

                }

            }

        }

    }

 do

 {
```

```
    Err = 0 ;
for (ii=0 ; ii < PartX ; ii++)
    {
        for (jj=0 ; jj < PartY ; jj++)
        {
            for (kk=0 ; kk < PartZ ; kk++)
            {
                ipi = ii+1;
                ini = ii-1;

                jpj = jj+1;
                jnj = jj-1;

                kpk = kk+1;
                knk = kk-1;

                if ( ipi  >= PartX )
                    {
                        ipi = 0 ;
                    }


                if ( ini  < 0 )
                    {
                        ini = PartX -1 ;
                    }

                if ( jpj  >= PartY )
                    {
                        jpj = 0 ;
                    }


                if ( jnj  < 0 )
                    {
                        jnj = PartY-1 ;
                    }

                if ( kpk  >= PartZ )
                    {
                        kpk = 0 ;
```

```
                    }


                if ( knk  < 0 )
                    {
                        knk = PartZ-1 ;
                    }
//-------------------------------------------------------------------------------------
----
            {
                Phi[ii][jj][kk] =
                    (((q*(DI[ii][jj][kk]-DE[ii][jj][kk])*dx*dx)/f)+
                    Phi[ipi][jj][kk]+Phi[ini][jj][kk]+
                    Phi[ii][jpj][kk]+Phi[ii][jnj][kk]+
                    Phi[ii][jj][kpk]+Phi[ii][jj][knk])/6.00 ;
    //-------------------------------------------------------------------------//
            /*  if(( ii >= 8) && (ii < 12) && ( jj >= 8) && (jj < 12) && ( kk >= 8) &&
(kk < 12) )
                {
                    Phi[ii][jj][kk] = 0.00;//Biase high voltage
                }*/

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

            }
//-------------------------------------------------------------------------------------
----
                Error[ii][jj][kk] =
                    100.00*(fabs((Phi[ii][jj][kk]-T_Phi[ii][jj][kk])/Phi[ii][jj][kk]));
                T_Phi[ii][jj][kk] = Phi[ii][jj][kk];
                    if ( Error[ii][jj][kk]  >= tolerate )
                    {
                        Er[ii][jj][kk] = 1 ;
                    }
                    else
                    {
                        Er[ii][jj][kk] = 0 ;
```

```
                    }

                    Err = Err + Er[ii][jj][kk] ;

                }

            }
        }
    }

while ( Err != 0) ;
//-----------------------------------------//

    TIPhi = IPhi;

    if( (step >= 500)&&(step < 1000)  || (step >= 1500)&&(step < 2000)  ||
        (step >= 2500)&&(step < 3000)
      )//BIAS //

                {

                    TIPhi = 0.00;
                }

    for (ii=8 ; ii < 12 ; ii++)

    {

        for (jj=8 ; jj < 12 ; jj++)

        {

            for (kk=8 ; kk < 12 ; kk++)

            {

                Phi[ii][jj][kk] = Phi[ii][jj][kk] + (TIPhi);//Biase high voltage

            }

        }

    }


    for (ii=0 ; ii < PartX ; ii++)

    {

        for (jj=0 ; jj < PartY ; jj++)

        {

            for (kk=0 ; kk < PartZ ; kk++)

            {

                DE[ii][jj][kk]=Density*exp((q*Phi[ii][jj][kk])/(Kb*TEMP));

            }

        }
```

```
    }

    for (ii=8 ; ii < 12 ; ii++)
    {
        for (jj=8 ; jj < 12 ; jj++)
        {
            for (kk=8 ; kk < 12 ; kk++)
            {
                DE[ii][jj][kk] = 0.00 ;//Biase high voltage
            }
        }
    }

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

void F_Electric()

{

long i,j,k;

for (i=0 ; i < (PartX) ; i++)

    {

    for (j=0 ; j < (PartY) ; j++)

        {

        for (k=0 ; k < (PartZ) ; k++)

            {


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

            if( (i == 0)&&(i != PartX-1)
                &&(j != 0)&&(j != PartY-1)
                &&(k != 0)&&(k != PartZ-1) )

                {

                    F_ElectricX(i,j,k,i+1,PartX-1);
                    F_ElectricY(i,j,k,j+1,j-1);
                    F_ElectricZ(i,j,k,k+1,k-1);//0jk

                }

            else if( (i != 0)&&(i != PartX-1)
                &&(j == 0)&&(j != PartY-1)
                &&(k != 0)&&(k != PartZ-1) )

                {

                    F_ElectricX(i,j,k,i+1,i-1);
                    F_ElectricY(i,j,k,j+1,PartY-1);
                    F_ElectricZ(i,j,k,k+1,k-1);//i0k

                }

            else if( (i != 0)&&(i != PartX-1)
                &&(j != 0)&&(j != PartY-1)
                &&(k == 0)&&(k != PartZ-1) )

                {
                    F_ElectricX(i,j,k,i+1,i-1);
                    F_ElectricY(i,j,k,j+1,j-1);
                    F_ElectricZ(i,j,k,k+1,PartZ-1); //ij0

                }

            else if( (i != 0)&&(i == PartX-1)
```

```
    &&(j != 0)&&(j != PartY-1)
    &&(k != 0)&&(k != PartZ-1))

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,k+1,k-1);//Ljk

    }

else if( (i != 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,k-1);//iLk

    }

else if( (i != 0)&&(i != PartX-1)
    &&(j != 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,0,k-1);//ijL

    }

else if( (i == 0)&&(i != PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k != 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,k+1,k-1);//00k

    }

else if( (i == 0)&&(i != PartX-1)
    &&(j != 0)&&(j != PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//0j0

    }
else if( (i != 0)&&(i != PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//i00
```

```
    }
else if( (i != 0)&&(i == PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k != 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,k+1,k-1);//L0k

    }
else if( (i != 0)&&(i == PartX-1)
    &&(j != 0)&&(j != PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//Lj0

    }
else if( (i != 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//iL0
    }
else if( (i == 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,k-1);//0Lk

    }
else if( (i == 0)&&(i != PartX-1)
    &&(j != 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,0,k-1);//0jL

    }


else if( (i != 0)&&(i != PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {
```

```
        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,0,k-1);//i0L

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,k-1);//LLk

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j != 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,j-1);
        F_ElectricZ(i,j,k,0,k-1);//LjL

    }

else if( (i != 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,0,k-1);//iLL

    }

else if( (i == 0)&&(i != PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//000

    }
else if( (i == 0)&&(i != PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,0,k-1);//00L

    }

else if( (i == 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
```

```
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//0L0

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//L00

    }

else if( (i == 0)&&(i != PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,i+1,PartX-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,0,k-1);//0LL

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k == 0)&&(k != PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,k+1,PartZ-1);//LL0

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j == 0)&&(j != PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,j+1,PartY-1);
        F_ElectricZ(i,j,k,0,k-1);//L0L

    }

else if( (i != 0)&&(i == PartX-1)
    &&(j != 0)&&(j == PartY-1)
    &&(k != 0)&&(k == PartZ-1) )

    {

        F_ElectricX(i,j,k,0,i-1);
        F_ElectricY(i,j,k,0,j-1);
        F_ElectricZ(i,j,k,0,k-1);//LLL
```

```
                }

            else

                {
                    F_ElectricX(i,j,k,i+1,i-1);
                    F_ElectricY(i,j,k,j+1,j-1);
                    F_ElectricZ(i,j,k,k+1,k-1);

                }


            }


        }

    }
return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double BXold[PartX][PartY][PartZ],BYold[PartX][PartY][PartZ],BZold[PartX][PartY][PartZ]
;

void F_Magnetic()

{

long mi,mj,mk;

for (mi=0 ; mi < (PartX) ; mi++)

    {

    for (mj=0 ; mj < (PartY) ; mj++)

        {

        for (mk=0 ; mk < (PartZ) ; mk++)

            {

            if( (mi == 0)&&(mj != 0)&&(mk != 0) )

                {

                    F_MagneticX(mi,mj,mk,mj-1,mk-1);
                    F_MagneticY(mi,mj,mk,mk-1,PartX-1);
                    F_MagneticZ(mi,mj,mk,PartX-1,mj-1);//0jk

                }

            else if( (mi != 0)&&(mj == 0)&&(mk != 0) )

                {

                    F_MagneticX(mi,mj,mk,PartY-1,mk-1);
                    F_MagneticY(mi,mj,mk,mk-1,mi-1);
                    F_MagneticZ(mi,mj,mk,mi-1,PartY-1);//i0k

                }

            else if( (mi != 0)&&(mj != 0)&&(mk == 0) )

                {
                    F_MagneticX(mi,mj,mk,mj-1,PartZ-1);
                    F_MagneticY(mi,mj,mk,PartZ-1,mi-1);
                    F_MagneticZ(mi,mj,mk,mi-1,mj-1);     //ij0

                }


            else if( (mi == 0)&&(mj == 0)&&(mk != 0) )

                {
```

```
                F_MagneticX(mi,mj,mk,PartY-1,mk-1);
                F_MagneticY(mi,mj,mk,mk-1,PartX-1);
                F_MagneticZ(mi,mj,mk,PartX-1,PartY-1);//00k

            }

        else if( (mi == 0)&&(mj != 0)&&(mk == 0) )

            {

                F_MagneticX(mi,mj,mk,mj-1,PartZ-1);
                F_MagneticY(mi,mj,mk,PartZ-1,PartX-1);
                F_MagneticZ(mi,mj,mk,PartX-1,mj-1);//0j0

            }
        else if( (mi != 0)&&(mj == 0)&&(mk == 0) )

            {

                F_MagneticX(mi,mj,mk,PartY-1,PartZ-1);
                F_MagneticY(mi,mj,mk,PartZ-1,mi-1);
                F_MagneticZ(mi,mj,mk,mi-1,PartY-1);//i00

            }


        else if( (mi == 0)&&(mj == 0)&&(mk == 0))

            {

                F_MagneticX(mi,mj,mk,PartY-1,PartZ-1);
                F_MagneticY(mi,mj,mk,PartZ-1,PartX-1);
                F_MagneticZ(mi,mj,mk,PartX-1,PartY-1);//000

            }


        else

            {
                F_MagneticX(mi,mj,mk,mj-1,mk-1);
                F_MagneticY(mi,mj,mk,mk-1,mi-1);
                F_MagneticZ(mi,mj,mk,mi-1,mj-1);

            }

        }

    }

for (mi=0 ; mi < (PartX) ; mi++)

    {

    for (mj=0 ; mj < (PartY) ; mj++)

        {

        for (mk=0 ; mk < (PartZ) ; mk++)

            {

                BXold[mi][mj][mk] = BX[mi][mj][mk] ;
```

```
                BYold[mi][mj][mk] = BY[mi][mj][mk];
                BZold[mi][mj][mk] = BZ[mi][mj][mk];

            }
        }
    }
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double TSIX[4][Particle],TSIY[4][Particle],TSIZ[4][Particle];

double RTSIX[Particle],RTSIY[Particle],RTSIZ[Particle];

double XI[4][Particle],XI1[4][Particle];
double YI[4][Particle],YI1[4][Particle];
double ZI[4][Particle],ZI1[4][Particle];

double XXI[Particle],XXI1[Particle];
double YYI[Particle],YYI1[Particle];
double ZZI[Particle],ZZI1[Particle];


double DI[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double Rho,LengthX,LengthY,LengthZ,t,IPhi;

double dx,dy,dz;


/*------------------------------------------------------------------------------ */
/*
/*-------------------------------------------------------------------------------*/

    long F_Eposition(double P,double L,double D)

    {

    long Eposition;

        if(P < 0)

        {

            P = L+fmod(P,L);

        Eposition = (long)floor(P/D);

        }

        else if(P >= L)

        {
            P = fmod(P,L);

        Eposition = (long)floor(P/D);

        }

        else

        {
```

```
            Eposition = (long)floor(P/D);
        }
    return (Eposition);
    }
/*------------------------------------------------------------------------------ */
/*
/*-----------------------------------------------------------------------------*/
    double F_TNposition(double TP,double TL,double TD)
    {
    double Tposition; ;
        if(TP < 0)
        {
            TP = TL+fmod(TP,TL);

        Tposition = TP;
        }
        else if(TP >= TL)
        {
            TP = fmod(TP,TL);
        Tposition = TP;
        }
        else
        {
            Tposition = TP;
        }
     return (Tposition);
    }
/*------------------------------------------------------------------------------ */
/*
/*-----------------------------------------------------------------------------*/

void Runge_Kutta()
{
long n;
/*------------------------------------------------------------------------------ */
/*Runge-Kutta Method */
/*-----------------------------------------------------------------------------*/
    for ( n = 0; n < Particle; n++)///1
    {
```

```
/*-------------------------------------------------------------------------------*/
/*ion
/*-------------------------------------------------------------------------------*/

        XI1[0][n] = t*(q/MI)*
        EX[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]+
        t*(q/MI)* YYI1[n]*
        BZ[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]-
        t*(q/MI)*ZZI1[n]*
        BY[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)];


        XI[0][n] =  t*(XXI1[n]+XI1[0][n]);

        RTSIX[n] = SIX[n] + (XI[0][n]);

        TSIX[0][n] = F_TNposition(RTSIX[n],LengthX,dx);

/*-------------------------------------------------------------------------------*/

        YI1[0][n] = t*(q/MI)*
        EY[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]+ t*(q/MI)*ZZI1[n]*
        BX[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]-
        t*(q/MI)*XXI1[n]*
        BZ[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)];

        YI[0][n] =  t*(YYI1[n]+YI1[0][n]);

        RTSIY[n] = SIY[n] + (YI[0][n]);

        TSIY[0][n] = F_TNposition(RTSIY[n],LengthY,dy);

/*-------------------------------------------------------------------------------*/

        ZI1[0][n] = t*(q/MI)*
        EZ[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]+
        t*(q/MI)*XXI1[n]*
        BY[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)]-
        t*(q/MI)*YYI1[n]*
        BX[F_Eposition(SIX[n],LengthX,dx)][F_Eposition(SIY[n],LengthY,dy)][F_Eposition(
SIZ[n],LengthZ,dz)];

        ZI[0][n] =  t*(ZZI1[n]+ZI1[0][n]);

        RTSIZ[n] = SIZ[n] + (ZI[0][n]);

        TSIZ[0][n] = F_TNposition(RTSIZ[n],LengthZ,dz);

/*-------------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------------*/

    }

    F_IDensity(TSIX[0],TSIY[0],TSIZ[0]);

    F_Phi();

    F_Electric();
```

```
/*------------------------------------------------------------------------------ */

/*------------------------------------------------------------------------------ */

for ( n = 0; n < Particle; n++)///2

{

/*-------------------------------------------------------------------------------*/
/*ion
/*-------------------------------------------------------------------------------*/

        XI1[1][n] = t*(q/MI)*
        EX[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]+
        t*(q/MI)*YI1[0][n]*
        BZ[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]-
        t*(q/MI)*ZI1[0][n]*
        BY[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)];

        XI[1][n] =  t*(XXI1[n]+(XI1[1][n]/2.00));

        RTSIX[n] = SIX[n] + (XI[1][n]/2.00);

        TSIX[1][n] = F_TNposition(RTSIX[n],LengthX,dx);

/*------------------------------------------------------------------------------ */

        YI1[1][n] = t*(q/MI)*
        EY[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]+
        t*(q/MI)*ZI1[0][n]*
        BX[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]-
        t*(q/MI)*XI1[0][n]*
        BZ[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)];


        YI[1][n] =  t*(YYI1[n]+(YI1[1][n]/2.00));

        RTSIY[n] = SIY[n] + (YI[1][n]/2.00);

        TSIY[1][n] = F_TNposition(RTSIY[n],LengthY,dy);

/*------------------------------------------------------------------------------ */

        ZI1[1][n] = t*(q/MI)*
        EZ[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]+
        t*(q/MI)*XI1[0][n]*
        BY[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)]-
        t*(q/MI)*YI1[0][n]*
        BX[F_Eposition(TSIX[0][n],LengthX,dx)][F_Eposition(TSIY[0][n],LengthY,dy)][F_Ep
osition(TSIZ[0][n],LengthZ,dz)];


        ZI[1][n] =  t*(ZZI1[n]+(ZI1[1][n]/2.00));

        RTSIZ[n] = SIZ[n] + (ZI[1][n]/2.00);

        TSIZ[1][n] = F_TNposition(RTSIZ[n],LengthZ,dz);
```

```
/*------------------------------------------------------------------------------ */

/*------------------------------------------------------------------------------ */

    }

    F_IDensity(TSIX[1],TSIY[1],TSIZ[1]);

    F_Phi();

    F_Electric();

/*------------------------------------------------------------------------------ */

/*------------------------------------------------------------------------------ */


for ( n = 0; n < Particle; n++)///3

{

/*-------------------------------------------------------------------------------*/
/*ion
/*-------------------------------------------------------------------------------*/

        XI1[2][n] = t*(q/MI)*
        EX[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]+
        t*(q/MI)*YI1[1][n]*
        BZ[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]-
        t*(q/MI)*ZI1[1][n]*
        BY[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)];

        XI[2][n] =  t*(XXI1[n]+(XI1[2][n]/2.00));

        RTSIX[n] = SIX[n] + (XI[2][n]/2.00);

        TSIX[2][n] = F_TNposition(RTSIX[n],LengthX,dx);

/*------------------------------------------------------------------------------ */

        YI1[2][n] = t*(q/MI)*
        EY[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]+
        t*(q/MI)*ZI1[1][n]*
        BX[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]-
        t*(q/MI)*XI1[1][n]*
        BZ[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)];


        YI[2][n] =  t*(YYI1[n]+(YI1[2][n]/2.00));

        RTSIY[n] = SIY[n] + (YI[2][n]/2.00);

        TSIY[2][n] = F_TNposition(RTSIY[n],LengthY,dy);

/*------------------------------------------------------------------------------ */

        ZI1[2][n] = t*(q/MI)*
        EZ[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]+
        t*(q/MI)*XI1[1][n]*
```

```
        BY[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)]-
        t*(q/MI)*YI1[1][n]*
        BX[F_Eposition(TSIX[1][n],LengthX,dx)][F_Eposition(TSIY[1][n],LengthY,dy)][F_Ep
osition(TSIZ[1][n],LengthZ,dz)];


        ZI[2][n] =  t*(ZZI1[n]+(ZI1[2][n]/2.00));

        RTSIZ[n] = SIZ[n] + (ZI[2][n]/2.00);

        TSIZ[2][n] = F_TNposition(RTSIZ[n],LengthZ,dz);

/*------------------------------------------------------------------------------ */

/*------------------------------------------------------------------------------ */

    }

    F_IDensity(TSIX[2],TSIY[2],TSIZ[2]);

    F_Phi();

    F_Electric();

/*------------------------------------------------------------------------------ */

/*------------------------------------------------------------------------------ */


for ( n = 0; n < Particle; n++)///4

{

/*-------------------------------------------------------------------------------*/
/*ion
/*-------------------------------------------------------------------------------*/

        XI1[3][n] = t*(q/MI)*
        EX[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]+
        t*(q/MI)*YI1[2][n]*
        BZ[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]-
        t*(q/MI)*ZI1[2][n]*
        BY[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)];


        XI[3][n] =  t*(XXI1[n]+(XI1[3][n]));

/*------------------------------------------------------------------------------ */

        YI1[3][n] = t*(q/MI)*
        EY[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]+
        t*(q/MI)*ZI1[2][n]*
        BX[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]-
        t*(q/MI)*XI1[2][n]*
        BZ[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)];

        YI[3][n] =  t*(YYI1[n]+(YI1[3][n]));

/*------------------------------------------------------------------------------ */
```

```
        ZI1[3][n] = t*(q/MI)*
        EZ[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]+
        t*(q/MI)*XI1[2][n]*
        BY[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)]-
        t*(q/MI)*YI1[2][n]*
        BX[F_Eposition(TSIX[2][n],LengthX,dx)][F_Eposition(TSIY[2][n],LengthY,dy)][F_Ep
osition(TSIZ[2][n],LengthZ,dz)];


        ZI[3][n] =  t*(ZZI1[n]+(ZI1[3][n]));
/*------------------------------------------------------------------------- */

    }

/*------------------------------------------------------------------------- */
/*------------------------------------------------------------------------- */
for ( n = 0; n < Particle; n++)
    {
        XXI1[n] = XXI1[n]+(1.00/6.00)*(XI1[0][n]+2.00*XI1[1][n]+2.00*XI1[2][n]+XI1[3][n
]);
        XXI[n] =  (1.00/6.00)*(XI[0][n]+2.00*XI[1][n]+2.00*XI[2][n]+XI[3][n]);
        SIX[n] = SIX[n] + XXI[n];
        SIX[n] =F_TNposition(SIX[n],LengthX,dx);
        YYI1[n] = YYI1[n]+(1.00/6.00)*(YI1[0][n]+2.00*YI1[1][n]+2.00*YI1[2][n]+YI1[3][n
]);
        YYI[n] =  (1.00/6.00)*(YI[0][n]+2.00*YI[1][n]+2.00*YI[2][n]+YI[3][n]);
        SIY[n] = SIY[n] + YYI[n];
        SIY[n] =F_TNposition(SIY[n],LengthY,dy);
        ZZI1[n] = ZZI1[n]+(1.00/6.00)*(ZI1[0][n]+2.00*ZI1[1][n]+2.00*ZI1[2][n]+ZI1[3][n
]);
        ZZI[n] =  (1.00/6.00)*(ZI[0][n]+2.00*ZI[1][n]+2.00*ZI[2][n]+ZI[3][n]);
        SIZ[n] = SIZ[n] + ZZI[n];
        SIZ[n] =F_TNposition(SIZ[n],LengthZ,dz);


    }
/*------------------------------------------------------------------------- */
/*------------------------------------------------------------------------- */
return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

char *C;

int  decimal, sign;

void F_PrintParticle()

{

long p;

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

FILE *fParticle;

    C = _fcvt((float)step+1.00,0,&decimal,&sign);

    strcat(C,"Particle.csv");

    fParticle = fopen(C,"w+t");

    if(fParticle == NULL)

    {
    printf("ERRorrrr-Phi Particle.csv !\n");

    exit(1);

    }

    else

    {

    fprintf(fParticle,"----------------------------%i----------------------------\n",
step);

    for(p = 0; p < Particle; p++)

    {
```

```
            {

                    {
                        fprintf(fParticle,"%i,%e,%e,%e\n",p,XXI[p],YYI[p],ZZI[p]);
                    }


            }
    }
    }
    fclose(fParticle);

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

char *A;

int  decimal, sign;

void F_PrintPHI()

{

long p;

long pi,pj,pk;

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

FILE *fError,*fPhi;


    for(p = 0; p < Particle; p++)

    {

        if ((SIX[p] < 0.00)|| (SIX[p] > LengthX) ||
            (SIY[p] < 0.00)|| (SIY[p] > LengthX) ||
            (SIZ[p] < 0.00|| SIZ[p] > LengthX))

            {

                fError = fopen("PError.txt","a+t");

                if(fError == NULL)

                    {
                        printf("ERRorrrrrrrrrrrrrrr-PError.txt !\n");

                        exit(1);
                    }

                else

                    {
```

```
                    fprintf(fError,"%i,%e,%e,%e\n",p,SIX[p],SIY[p],SIZ[p]);
                    fclose(fError);
                }
            }
        }
        for(p = 0; p < Particle; p++)
        {
            if ((SIX[p] < 0.00)|| (SIX[p] > LengthX) ||
                (SIY[p] < 0.00)|| (SIY[p] > LengthX) ||
                (SIZ[p] < 0.00|| SIZ[p] > LengthX))

            {
                printf("Errorrrr from POSITION !");
                exit(0);
            }
        }

//-------------------------------------------------------------------------//
{
    A = _fcvt((float)step+1.00,0,&decimal,&sign);

    strcat(A,".csv");
    fPhi = fopen(A,"w+t");

if(fPhi == NULL)
{
    printf("ERRorrrr-Phi record.csv !\n");
    exit(1);
}
else
{
    fprintf(fPhi,"----------------------------%i----------------------------\n",step)
;
    for(pi = 0; pi < PartX; pi++)
    {
        for(pj =0 ; pj < PartY; pj++)
        {
```

```
            for(pk =0 ; pk < PartY; pk++)
            {
            fprintf(fPhi,"%i,%i,%i,%e,%e,%e\n",pi,pj,pk,Phi[pi][pj][pk],DI[pi][pj][pk],
DE[pi][pj][pk]);
            }
        }
    }
}
fprintf(fPhi,"-------------------------------------------------------------\n");
fclose(fPhi);
}

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "constant.h"

double SIX[Particle],SIY[Particle],SIZ[Particle];

double DI[PartX][PartY][PartZ],DE[PartX][PartY][PartZ];

double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

double BX[PartX][PartY][PartZ],BY[PartX][PartY][PartZ],BZ[PartX][PartY][PartZ];

double XXI[Particle],XXI1[Particle];

double YYI[Particle],YYI1[Particle];

double ZZI[Particle],ZZI1[Particle];

double Rho,LengthX,LengthY,LengthZ,t,IPhi,TEMP,dx,dy,dz,TIME_STEP;

long step;

long AOH ;

void F_PrintNormal()

{

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

FILE *floveAOH,*floveAOHphi,*floveAOHna;;


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

{
    floveAOH=fopen("LoveAoh.csv","a+t");
    floveAOHphi=fopen("LoveAohphi.csv","a+t");
    floveAOHna=fopen("LoveAohNAA.csv","a+t");

if((floveAOH == NULL)||(floveAOHphi== NULL)||(floveAOHna == NULL))

    {

        printf("ERRorrrr-Love AOH.csv !\n");

        exit(1);

    }


else

{

    for( AOH=0 ; AOH < PartX ; AOH++)
```

```
        {

            fprintf(floveAOH,"%e,",DI[AOH][9][9],DE[AOH][9][9]);
            fprintf(floveAOHphi,"%e,",Phi[AOH][9][9]);

        }

        fprintf(floveAOH,"\n");
        fprintf(floveAOHphi,"\n");

    for(AOH = 0; AOH < Particle; AOH ++)

    {

                if( (SIX[AOH] >= 7.00*dx) && (SIX[AOH] < 13.00*dx) &&
                    (SIY[AOH] >= 7.00*dy) && (SIY[AOH] < 13.00*dy) &&
                    (SIZ[AOH] >= 7.00*dz) && (SIZ[AOH] < 13.00*dz) )

                {

                    fprintf(floveAOHna,"%i,%i,%e,%e,%e,%e,%e,%e\n",step,AOH,XXI1[AOH],Y
YI1[AOH],ZZI1[AOH],SIX[AOH],SIY[AOH],SIZ[AOH]);

                }

    }


}

fclose(floveAOH);
fclose(floveAOHphi);
fclose(floveAOHna);

}

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double Phi[PartX][PartY][PartZ];

double IPhi;

void ClearPhi()

{

 long ii,jj,kk;

    for (ii=0 ; ii < PartX ; ii++)

    {

        for (jj=0 ; jj < PartY ; jj++)

        {

            for (kk=0 ; kk < PartZ ; kk++)

            {

                //Phi[ii][jj][kk] = 0.00;

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

                if( (ii == 0)&&(ii != PartX-1)
                &&(jj != 0)&&(jj != PartY-1)
                &&(kk != 0)&&(kk != PartZ-1) )

                {

                    Phi[ii][jj][kk] = 0.00;//0jk

                }

            else if( (ii != 0)&&(ii != PartX-1)
                &&(jj == 0)&&(jj != PartY-1)
                &&(kk != 0)&&(kk != PartZ-1) )

                {

                    Phi[ii][jj][kk] = 0.00;//i0k

                }

            else if( (ii != 0)&&(ii != PartX-1)
                &&(jj != 0)&&(jj != PartY-1)
                &&(kk == 0)&&(kk != PartZ-1) )

                {
                    Phi[ii][jj][kk] = 0.00;//ij0

                }

            else if( (ii != 0)&&(ii == PartX-1)
                &&(jj != 0)&&(jj != PartY-1)
                &&(kk != 0)&&(kk != PartZ-1))

                {
```

```
        Phi[ii][jj][kk] = 0.00;//Ljk

    }

else if( (ii != 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk != 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//iLk

    }

else if( (ii != 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//ijL

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//00k

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj != PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//0j0

    }
else if( (ii != 0)&&(ii != PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//i00

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//L0k

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj != 0)&&(jj != PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {
```

```
        Phi[ii][jj][kk] = 0.00;//Lj0

    }

else if( (ii != 0)&&(ii!= PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//iL0
    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk != 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//0Lk

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//0jL

    }



else if( (ii != 0)&&(ii != PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//i0L

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk!= 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//LLk

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj!= 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//LjL

    }

else if( (ii != 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
```

```
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//iLL

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//000

    }
else if( (ii == 0)&&(ii != PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//00L

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//0L0

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj == 0)&&(jj != PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//L00

    }

else if( (ii == 0)&&(ii != PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk != 0)&&(kk == PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//0LL

    }

else if( (ii != 0)&&(ii == PartX-1)
    &&(jj != 0)&&(jj == PartY-1)
    &&(kk == 0)&&(kk != PartZ-1) )

    {

        Phi[ii][jj][kk] = 0.00;//LL0

    }

else if( (ii != 0)&&(ii == PartX-1)
```

```
            &&(jj == 0)&&(jj != PartY-1)
            &&(kk != 0)&&(kk == PartZ-1) )

            {

                Phi[ii][jj][kk] = 0.00;//L0L

            }

        else if( (ii != 0)&&(ii == PartX-1)
            &&(jj != 0)&&(jj == PartY-1)
            &&(kk!= 0)&&(kk == PartZ-1) )

            {

                Phi[ii][jj][kk] = 0.00;//LLL

            }

        else

            {
                Phi[ii][jj][kk] = 0.00;

            }
        //-----------------------------------------------------//

        }

    }

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double Phi[PartX][PartY][PartZ];

double EX[PartX][PartY][PartZ];

double dx,dy,dz,t;

void F_ElectricX(long xii,long xjj,long xkk,long xiiPlus,long xiiNeg)

{


                EX[xii][xjj][xkk]=
                    (Phi[xiiNeg][xjj][xkk] - Phi[xiiPlus][xjj][xkk])/(2.00*dx);


return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double Phi[PartX][PartY][PartZ];

double EY[PartX][PartY][PartZ];

double dx,dy,dz,t;

void F_ElectricY(long yii,long yjj,long ykk,long yjjPlus,long yjjNeg)

{


                EY[yii][yjj][ykk]=
                    (Phi[yii][yjjNeg][ykk] - Phi[yii][yjjPlus][ykk])/(2.00*dy);


return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double Phi[PartX][PartY][PartZ];

double EZ[PartX][PartY][PartZ];

double dx,dy,dz,t;

void F_ElectricZ(long zii,long zjj,long zkk,long zkkPlus,long zkkNeg)

{


                EZ[zii][zjj][zkk]=
                    (Phi[zii][zjj][zkkNeg] - Phi[zii][zjj][zkkPlus])/(2.00*dz);

return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double BX[PartX][PartY][PartZ];

double BXold[PartX][PartY][PartZ];

double dx,dy,dz,t;

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

void F_MagneticX(long mxi,long mxj,long mxk,long mzjNeg, long mykNeg)

{


                BX[mxi][mxj][mxk]=

                    BXold[mxi][mxj][mxk]-
                    t*(((EZ[mxi][mxj][mxk] - EZ[mxi][mzjNeg][mxk])/(dy))-
                    ((EY[mxi][mxj][mxk] - EY[mxi][mxj][mykNeg])/(dz)));

return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double BY[PartX][PartY][PartZ];

double BYold[PartX][PartY][PartZ];

double dx,dy,dz,t;

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

void F_MagneticY(long myi,long myj,long myk,long mxkNeg,long mziNeg)

{


                BY[myi][myj][myk]=

                    BYold[myi][myj][myk]-
                    t*(((EX[myi][myj][myk] - EX[myi][myj][mxkNeg])/(dz))-
                    ((EZ[myi][myj][myk] - EZ[mziNeg][myj][myk])/(dx)));

return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "constant.h"


double BZ[PartX][PartY][PartZ];

double BZold[PartX][PartY][PartZ];

double dx,dy,dz,t;

double EX[PartX][PartY][PartZ],EY[PartX][PartY][PartZ],EZ[PartX][PartY][PartZ];

void F_MagneticZ(long mzi,long mzj,long mzk,long myiNeg,long mxjNeg)

{


                BZ[mzi][mzj][mzk]=

                    BZold[mzi][mzj][mzk]-
                    t*(((EY[mzi][mzj][mzk] - EY[myiNeg][mzj][mzk])/(dx))-
                    ((EX[mzi][mzj][mzk] - EX[mzi][mxjNeg][mzk])/(dy)));
return;
}
```

APPENDIX B

The Input files and codes for ion coating and implantation

LengthX:
4.449E-09
LengthY:
4.449E-09
LengthZ:
4.449E-09
Number1X:
11
Number1Y:
11
Number1Z:
11
Particle
80
Number of Proton1:
26.00
Mass Number1:
55.845
Number of Proton2:
78.00
Mass Number2:
195.078
Density:
7.874
INTERVAL TIME:
1.00e-16
TIME STEP:
4000
ION:
1.00
ION Position X:
2.1233864E-09
ION Position Y:
2.1233864E-09
ION Position Z:
-1.222250E-09
ION Velocity X:
0.00
ION Velocity Y:
0.00
ION Velocity Z:
28634.21077

----------Input file----------

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

double f_CoulombF(double CouQ1,double CouQ2, double CouMI,double CouMX,double CouMY,dou
ble CouMZ,double CoatZ,double BaseZ)

{
    double fzero,PI;
    double CoulombF ;
    double CoulombPo,DCoulombPo;
    double ScreenF,DScreenF;
    double aa;

    fzero = 8.850e-12;

    PI = 3.14;

    if( (CouMX == 0.00) && (CouMY == 0.00)&& (CouMZ == 0.00))

    {
        CoulombF =0.00;
    }

    else

    {
        aa = (4.60408E-11)/(pow(CoatZ,0.23)+pow(BaseZ,0.23)) ;

        CoulombPo = ((CouQ1*CouQ2)/(4*PI*fzero))*pow((pow(CouMX,2)+pow(CouMY,2)+pow(Cou
MZ,2)),-0.5);

        DCoulombPo = ((CouQ1*CouQ2)/(4*PI*fzero))*CouMI*pow((pow(CouMX,2)+pow(CouMY,2)+
pow(CouMZ,2)),-1.5);

        ScreenF = 0.1818*exp(-3.2*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0.5)/aa
))
                +0.5099*exp(-0.9423*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0.5)/
aa))
                +0.2802*exp(-0.4029*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0.5)/
aa))
                +0.02817*exp(-0.2016*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0.5)
/aa));

        DScreenF = (-3.2*CouMI*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),1.5)/aa))*
                        0.1818*exp(-3.2*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0
.5)/aa))
                +(-0.9423*CouMI*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),1.5)/aa))
*
                        0.5099*exp(-0.9423*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)
),0.5)/aa))
                +(-0.4029*CouMI*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),1.5)/aa))
*
                        0.2802*exp(-0.4029*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)
),0.5)/aa))
                +(-0.2016*CouMI*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),1.5)/aa))
*
                0.02817*exp(-0.2016*(pow((pow(CouMX,2)+pow(CouMY,2)+pow(CouMZ,2)),0.5)/
aa));


        CoulombF = CoulombPo*DScreenF + ScreenF*DCoulombPo;
    }

return(CoulombF);

}
```

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

double f_LJF(double LJMI,double LJMX,double LJMY,double LJMZ)

{
    double f,zigma;
    double LJF ;

    f= 4.15E-17;

    zigma=2.33925e-10;

    if( (LJMX == 0.00) && (LJMY == 0.00)&& (LJMZ == 0.00))
    {
        LJF =0.00;
    }

    else

    {

        LJF = (48*f*pow(zigma,12)*LJMI*pow((pow(LJMX,2)+pow(LJMY,2)+pow(LJMZ,2)),-7))
                    -(24*f*pow(zigma,6) *LJMI*pow((pow(LJMX,2)+pow(LJMY,2)+pow(LJMZ,2)),
-4));

    }

return(LJF);

}

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

double f_TForce(double ***FFF,long NNX,long NNY,long NNZ)

{
    double TForce;

    long tfi,tfj,tfk;

    TForce = 0.00;

    for( tfi = 0 ; tfi < NNX ; tfi++)
    {
     for( tfj = 0 ; tfj < NNY ; tfj++)
     {
      for( tfk = 0 ; tfk < NNZ ; tfk++)
      {

        TForce = TForce + FFF[tfi][tfj][tfk];

      }
     }
    }

return(TForce);

}

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

```
double f_ITForce(double *IFF,long IIN)

{
    double ITForce;

    long tf;

    ITForce = 0.00;

    if( IIN == 0 )
    {

        ITForce = 0.00;
    }

    else
    {

      for( tf = 0 ; tf < IIN ; tf++)

      {

        ITForce = ITForce + IFF[tf];

      }
    }

return(ITForce);

}

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

double f_TPosition(double TION,double PBASE)
{
    double RRR;

        RRR = TION-PBASE;

        //printf("%e,%e,\n",TION,PBASE);

    return(RRR);

}

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

double F_CheckPosition(double TP,double TL)
{


    double Tposition; ;

        if(TP < 0)

        {
            TP = TL+fmod(TP,TL);


        Tposition = TP;

        }

        else if(TP >= TL)
```

```
        {
            TP = fmod(TP,TL);
        Tposition = TP;
        }
        else
        {
            Tposition = TP;
        }
        //printf("%e\n",Tposition);
    //Tposition = TP;

    return (Tposition);
}
//-------------------------------------------------------------------------------------------
--//
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

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

long step,TIME_STEP ;

long aoh;

double IonX,IonY,IonZ;

double IXX1,IYY1,IZZ1 ;

double LengthX,LengthY,LengthZ;

long Particle, NParticle;

void main()

{

F_Input();

F_Memory();

F_ChargeDensity();

F_File();

for( aoh= NParticle ; aoh < Particle ; aoh++ )
{

    step = 0;

    F_Start();

    do{
    step = step + 1;
    F_SPrint();
    F_BaseIonForce();
    F_Print();

    }while (step <= TIME_STEP);

    F_Particle();

}


F_Free();

return;


}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

long TIME_STEP ;

double LengthX,LengthY,LengthZ;

double Proton,Mass,Density,ADensity,t;

double ProtonCoat,MassCoat;

double dx,dy,dz;

long N1X,N1Y,N1Z;

long Particle;

double InIon ;

double InPX,InPY,InPZ;

double InVX,InVY,InVZ;

void F_Input()

{

FILE *finput,*foutput;

finput = fopen("InPut.txt","r+t");


if(finput == NULL)

{
    printf("ERRorrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr !\n");

    exit(1);

}

else

{


    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthX);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthY);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&LengthZ);

    fseek(finput,9L, SEEK_CUR);
    fscanf(finput,"%ld\n",&N1X);

    fseek(finput,9L, SEEK_CUR);
    fscanf(finput,"%ld\n",&N1Y);

    fseek(finput,9L, SEEK_CUR);
    fscanf(finput,"%ld\n",&N1Z);

    fseek(finput,10L, SEEK_CUR);
    fscanf(finput,"%ld\n",&Particle);
```

```
    fseek(finput,19L, SEEK_CUR);
    fscanf(finput,"%lg\n",&Proton);

    fseek(finput,13L, SEEK_CUR);
    fscanf(finput,"%lg\n",&Mass);

    fseek(finput,19L, SEEK_CUR);
    fscanf(finput,"%lg\n",&ProtonCoat);

    fseek(finput,13L, SEEK_CUR);
    fscanf(finput,"%lg\n",&MassCoat);

    fseek(finput,8L, SEEK_CUR);
    fscanf(finput,"%lg\n",&Density);

    fseek(finput,14L, SEEK_CUR);
    fscanf(finput,"%lg\n",&t);

    fseek(finput,10L, SEEK_CUR);
    fscanf(finput,"%ld\n",&TIME_STEP);

    fseek(finput,4L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InIon);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InPX);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InPY);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InPZ);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InVX);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InVY);

    fseek(finput,15L, SEEK_CUR);
    fscanf(finput,"%lg\n",&InVZ);



}
fclose(finput);

    dx = LengthX/N1X;

    dy = LengthY/N1Y;

    dz = LengthZ/N1Z;

ADensity = (Density/Mass)*6.02e23 ;


foutput = fopen("c_data.txt","w+");


if(foutput == NULL)

{
    printf("ERRorrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr !\n");
```

```
    exit(1);
}
else
{
    fprintf(foutput,"Number of Proton  %e\n",Proton);
    fprintf(foutput,"Mass Number  %e\n",Mass);
    fprintf(foutput,"Density  %e\n",Density);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Number of Protron material Coating  %f\n",ProtonCoat);
    fprintf(foutput,"Mass Number of material Coating  %f\n",MassCoat);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Atom Density  %e\n",ADensity);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"LengthX %e \nLengthY %e \nLengthZ %e\n",LengthX,LengthY,LengthZ);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"PartX %d \nPartY %d \nPartZ %d\n",N1X,N1Y,N1Z);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Number of Coating Particles  %d \n",Particle);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Interval Time %e \nTime STEP %d\n",t,TIME_STEP);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Ion Charge  %f \n",InIon);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Ion Starting Position  %e,%e,%e \n",InPX,InPY,InPZ);
    fprintf(foutput,"----------------------------\n");
    fprintf(foutput,"Ion Velocity  %e,%e,%e \n",InVX,InVY,InVZ);
    fprintf(foutput,"----------------------------\n");
}
fclose(foutput);
//exit(0);
return;
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

double ***Q1,***CQ1 ;

double ***FX1,***FY1,***FZ1;

double ***TFX1,***TFY1,***TFZ1;

double ***FCouX1,***FCouY1,***FCouZ1;

double ***FLJX1,***FLJY1,***FLJZ1;

double ***TFCouX1,***TFCouY1,***TFCouZ1;

double ***TFLJX1,***TFLJY1,***TFLJZ1;

double ***RX1,***RY1,***RZ1,***R1;

double ***TRX1,***TRY1,***TRZ1 ;

double ***B1X,***B1Y,***B1Z;

double ***TBX1,***TBY1,***TBZ1;

//-------------------------
double *Qp,*CQp ;

double *FXp,*FYp,*FZp;

double *TFXp,*TFYp,*TFZp;

double *FCouXp,*FCouYp,*FCouZp;

double *FLJXp,*FLJYp,*FLJZp;

double *TFCouXp,*TFCouYp,*TFCouZp;

double *TFLJXp,*TFLJYp,*TFLJZp;

double *RXp,*RYp,*RZp,*Rp;

double *TRXp,*TRYp,*TRZp ;

double *BpX,*BpY,*BpZ;

double *TBXp,*TBYp,*TBZp;
//-------------------------

long N1X,N1Y,N1Z;

long Particle;

void F_Memory()

{

long ix,iy,iz;

//-------------------------------------------------------------------------------------
----
Q1 =   malloc( N1X * sizeof( double ) );
CQ1 =  malloc( N1X * sizeof( double ) );

FX1 =   malloc( N1X * sizeof( double ) );
FY1 =   malloc( N1X * sizeof( double ) );
```

```
FZ1 =   malloc( N1X * sizeof( double ) );

FCouX1 =   malloc( N1X * sizeof( double ) );
FCouY1 =   malloc( N1X * sizeof( double ) );
FCouZ1 =   malloc( N1X * sizeof( double ) );

FLJX1 =   malloc( N1X * sizeof( double ) );
FLJY1 =   malloc( N1X * sizeof( double ) );
FLJZ1 =   malloc( N1X * sizeof( double ) );

TFX1 =   malloc( N1X * sizeof( double ) );
TFY1 =   malloc( N1X * sizeof( double ) );
TFZ1=   malloc( N1X * sizeof( double ) );

TFCouX1 =   malloc( N1X * sizeof( double ) );
TFCouY1 =   malloc( N1X * sizeof( double ) );
TFCouZ1 =   malloc( N1X * sizeof( double ) );

TFLJX1 =   malloc( N1X * sizeof( double ) );
TFLJY1 =   malloc( N1X * sizeof( double ) );
TFLJZ1 =   malloc( N1X * sizeof( double ) );

RX1 =   malloc( N1X * sizeof( double ) );
RY1 =   malloc( N1X * sizeof( double ) );
RZ1 =   malloc( N1X * sizeof( double ) );
R1  =   malloc( N1X * sizeof( double ) );

TRX1 =   malloc( N1X * sizeof( double ) );
TRY1 =   malloc( N1X * sizeof( double ) );
TRZ1 =   malloc( N1X * sizeof( double ) );

B1X =   malloc( N1X * sizeof( double ) );
B1Y =   malloc( N1X * sizeof( double ) );
B1Z =   malloc( N1X * sizeof( double ) );

TBX1 =   malloc( N1X * sizeof( double ) );
TBY1 =   malloc( N1X * sizeof( double ) );
TBZ1 =   malloc( N1X * sizeof( double ) );

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

for ( ix = 0 ; ix <N1X ; ix++)
{
        Q1[ix]  =  malloc( N1Y * sizeof( double ) );
        CQ1[ix]  =  malloc( N1Y * sizeof( double ) );

        FX1[ix]  =  malloc( N1Y * sizeof( double ) );
        FY1[ix]  =  malloc( N1Y * sizeof( double ) );
        FZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        FCouX1[ix]  =  malloc( N1Y * sizeof( double ) );
        FCouY1[ix]  =  malloc( N1Y * sizeof( double ) );
        FCouZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        FLJX1[ix]  =  malloc( N1Y * sizeof( double ) );
        FLJY1[ix]  =  malloc( N1Y * sizeof( double ) );
        FLJZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        TFX1[ix]  =  malloc( N1Y * sizeof( double ) );
        TFY1[ix]  =  malloc( N1Y * sizeof( double ) );
        TFZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        TFCouX1[ix]  =  malloc( N1Y * sizeof( double ) );
        TFCouY1[ix]  =  malloc( N1Y * sizeof( double ) );
        TFCouZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        TFLJX1[ix]  =  malloc( N1Y * sizeof( double ) );
        TFLJY1[ix]  =  malloc( N1Y * sizeof( double ) );
```

```
        TFLJZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        RX1[ix]  =  malloc( N1Y * sizeof( double ) );
        RY1[ix]  =  malloc( N1Y * sizeof( double ) );
        RZ1[ix]  =  malloc( N1Y * sizeof( double ) );
        R1[ix]   =  malloc( N1Y * sizeof( double ) );

        TRX1[ix]  =  malloc( N1Y * sizeof( double ) );
        TRY1[ix]  =  malloc( N1Y * sizeof( double ) );
        TRZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        B1X[ix]  =  malloc( N1Y * sizeof( double ) );
        B1Y[ix]  =  malloc( N1Y * sizeof( double ) );
        B1Z[ix]  =  malloc( N1Y * sizeof( double ) );

        TBX1[ix]  =  malloc( N1Y * sizeof( double ) );
        TBY1[ix]  =  malloc( N1Y * sizeof( double ) );
        TBZ1[ix]  =  malloc( N1Y * sizeof( double ) );

        //-------------------------------------------------------
}

for ( ix = 0 ; ix <N1X ; ix++)
for ( iy = 0 ; iy <N1Y ; iy++)
{
{
        Q1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        CQ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        FX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        FCouX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FCouY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FCouZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        FLJX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FLJY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        FLJZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        TFX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        TFCouX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFCouY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFCouZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        TFLJX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFLJY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TFLJZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        RX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        RY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        RZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        R1[ix][iy]  =  malloc( N1Z * sizeof( double ) );

        TRX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TRY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TRZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

        B1X[ix][iy] =  malloc( N1Z * sizeof( double ) );
        B1Y[ix][iy] =  malloc( N1Z * sizeof( double ) );
        B1Z[ix][iy] =  malloc( N1Z * sizeof( double ) );

        TBX1[ix][iy] =  malloc( N1Z * sizeof( double ) );
        TBY1[ix][iy] =  malloc( N1Z * sizeof( double ) );
```

```
        TBZ1[ix][iy] =  malloc( N1Z * sizeof( double ) );

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

}
}

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

Qp =   malloc( Particle * sizeof( double ) );
CQp =  malloc( Particle * sizeof( double ) );

FXp =   malloc( Particle * sizeof( double ) );
FYp =   malloc( Particle * sizeof( double ) );
FZp =   malloc( Particle * sizeof( double ) );

FCouXp =   malloc( Particle * sizeof( double ) );
FCouYp =   malloc( Particle * sizeof( double ) );
FCouZp =   malloc( Particle * sizeof( double ) );

FLJXp =   malloc( Particle * sizeof( double ) );
FLJYp =   malloc( Particle * sizeof( double ) );
FLJZp =   malloc( Particle * sizeof( double ) );

TFXp =   malloc( Particle * sizeof( double ) );
TFYp =   malloc( Particle * sizeof( double ) );
TFZp=   malloc( Particle * sizeof( double ) );

TFCouXp =   malloc( Particle * sizeof( double ) );
TFCouYp =   malloc( Particle * sizeof( double ) );
TFCouZp =   malloc( Particle * sizeof( double ) );

TFLJXp =   malloc( Particle * sizeof( double ) );
TFLJYp =   malloc( Particle * sizeof( double ) );
TFLJZp =   malloc( Particle * sizeof( double ) );

RXp =   malloc( Particle * sizeof( double ) );
RYp =   malloc( Particle * sizeof( double ) );
RZp =   malloc( Particle * sizeof( double ) );
Rp  =   malloc( Particle * sizeof( double ) );

TRXp =   malloc( Particle * sizeof( double ) );
TRYp =   malloc( Particle * sizeof( double ) );
TRZp =   malloc( Particle * sizeof( double ) );

BpX =   malloc( Particle * sizeof( double ) );
BpY =   malloc( Particle * sizeof( double ) );
BpZ =   malloc( Particle * sizeof( double ) );

TBXp =   malloc( Particle * sizeof( double ) );
TBYp =   malloc( Particle * sizeof( double ) );
TBZp =   malloc( Particle * sizeof( double ) );


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

return;

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>

double ***Q1,***CQ1;

double *Qp,*CQp ;

double CIon ,InIon ;

double ADensity,Proton,ProtonCoat;

long N1X,N1Y,N1Z;

long Particle;

double dx,dy,dz;

double MI,Mass;

double MassCoat;

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

void F_ChargeDensity()
{

    long QI,QJ,QK ;

    MI = MassCoat*1.660574e-27;

    CIon = InIon*1.66e-19 ;

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

    for(QI = 0 ; QI < N1X ;QI++)

    {

        for(QJ = 0 ; QJ < N1Y ;QJ++)

        {

            for(QK = 0 ; QK < N1Z ;QK++)

            {

                Q1[QI][QJ][QK] = 1.00;

                CQ1[QI][QJ][QK] = 1.66e-19*Proton*Q1[QI][QJ][QK];//proton//

            }


        }

    }

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

    for(QI = 0 ; QI < Particle ;QI++)

            {

                Qp[QI] = 1.00;
```

```
            CQp[QI] = 1.66e-19*ProtonCoat*Qp[QI];//Ion proton//
        }
//-------------------------------------------------------------//
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>


double *BpX,*BpY,*BpZ;

double *ATime,*XP,*YP,*ZP,*VXP,*VYP,*VZP;

double AT,AX,AY,AZ,AVX,AVY,AVZ;

long rr,NData,StDATA;

long Particle,NParticle;

char C[40];



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

void F_File()

{
//-------------------------------------------------------------------------------------------
----


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

    FILE *Data;
    FILE *File;
    FILE *BPrint;

    File = fopen("File.txt","r+t");

if(File == NULL)

{
    printf("ERRorrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr !\n");

    exit(1);

}

else

{

    fseek(File,14L, SEEK_CUR);
    fscanf(File,"%ld\n",&StDATA);

    fseek(File,15L, SEEK_CUR);
    fscanf(File,"%ld\n",&NData);



}

fclose(File);
```

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

    for( rr = StDATA ; rr < NData ; rr++)

    {

        sprintf(C,"%d",rr);

        strcat(C,"Data.csv");

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

        Data = fopen(C,"r");

        BPrint = fopen("data1.csv","a+t");

        if(Data == NULL)
        {

            printf("Can not Open DATA \n");
            exit(0);
        }

        else
        {

            do
            {

            fscanf(Data,"%lg,%lg,%lg,%lg,%lg,%lg,%lg,\n",&AT,&AX,&AY,&AZ,&AVX,&AVY,&AVZ
) ;

            }while ((fscanf(Data,"%lg,%lg,%lg,%lg,%lg,%lg,%lg,\n",&AT,&AX,&AY,&AZ,&AVX,
&AVY,&AVZ))!=EOF);

            fclose(Data);


        BpX[rr] = AX ;
        BpY[rr] = AY ;
        BpZ[rr] = AZ ;


fprintf(BPrint,"%d,%e,%e,%e,%e,%e,%e,%e,\n",rr,AT,AX,AY,AZ,AVX,AVY,AVZ);
fclose(BPrint);

printf("%d\n",rr);

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


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

    }

}


NParticle = NData;

return;

}
```

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

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

//-------------------------------------------------------------------------------------------
----
double ***B1X,***B1Y,***B1Z;

double *BpX,*BpY,*BpZ;

double dx,dy,dz;

double IonX,IonY,IonZ;

long N1X,N1Y,N1Z;

long Particle,NParticle;

double LengthX,LengthY,LengthZ;

double IXX1,IYY1,IZZ1 ;

double InPX,InPY,InPZ;

double InVX,InVY,InVZ;

long aoh;

void F_Start()

{

long si,sj,sk;

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


    IonX = InPX+(2.5278409E-12)*aoh;
    IonY = InPY+(2.5278409E-12)*aoh;
    IonZ = InPZ;

    IXX1 = InVX;
    IYY1 = InVY;
    IZZ1 = InVZ;

    //printf("%e,%e,%e,%e,%e,%e\n",IonX,IonY,IonZ,IXX1,IYY1,IZZ1);

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

// SIMPLE CUBIC (SC)

for (si=0 ; si < N1X ; si ++)

    {

        for (sj=0 ; sj < N1Y ; sj ++)

        {

            for (sk=0 ; sk < N1Z ; sk ++)

            {

                B1X[si][sj][sk] = si*dx;
                B1Y[si][sj][sk] = sj*dy;
```

```
                B1Z[si][sj][sk] = sk*dz;

                //printf("%i,%i,%i,%e,%e,%e\n",si,sj,sk,B1X[si][sj][sk],B1Y[si][sj][sk]
,B1Z[si][sj][sk]);

            }
        }
    }

//exit(0);

/*for (si=0 ; si < NParticle ; si ++)

            {

                BpX[si] = 0.00;
                BpY[si] = 0.00;
                BpZ[si] = 0.00;

                //printf("%i,%i,%i,%e,%e,%e\n",si,sj,sk,B1X[si][sj][sk],B1Y[si][sj][sk]
,B1Z[si][sj][sk]);

            }
*/
return;

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>

double IXX1,IYY1,IZZ1 ;

double IXX,IYY,IZZ ;

double IonX,IonY,IonZ;

double t;

char C[40];

int  decimal, sign;

long aoh;

long step;
//-------------------------------------------------------------------------------------
--//

void F_SPrint()
{
    long spti,sptj,sptk;

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

FILE *fSData;

    if(step == 0)
    {

    sprintf(C,"%d",aoh);

    strcat(C,"Data.csv");

    fSData = fopen(C,"a+t");

    if(fSData == NULL)

    {
    printf("ERRorrrr-Phi Particle.csv !\n");

    exit(1);

    }

    else

    {

    fprintf(fSData,"%e,%e,%e,%e,%e,%e,%e,\n",t*step,IonX,IonY,IonZ,IXX1,IYY1,IZZ1);

    }

    fclose(fSData);

    //printf("%e,%e,%e,%e,%e,%e,%e,\n",t*step,IonX,IonY,IonZ,IXX1,IYY1,IZZ1);
    }
//-------------------------------------------------------------------------------------
--//
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "function.h"

double ***Q1,***CQ1 ;

double ***FX1,***FY1,***FZ1;

double ***TFX1,***TFY1,***TFZ1;

double ***FCouX1,***FCouY1,***FCouZ1;

double ***FLJX1,***FLJY1,***FLJZ1;

double ***TFCouX1,***TFCouY1,***TFCouZ1;

double ***TFLJX1,***TFLJY1,***TFLJZ1;

double ***RX1,***RY1,***RZ1,***R1;

double ***TRX1,***TRY1,***TRZ1 ;

double ***B1X,***B1Y,***B1Z;

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

double *Qp,*CQp ;

double *FXp,*FYp,*FZp;

double *TFXp,*TFYp,*TFZp;

double *FCouXp,*FCouYp,*FCouZp;

double *FLJXp,*FLJYp,*FLJZp;

double *TFCouXp,*TFCouYp,*TFCouZp;

double *TFLJXp,*TFLJYp,*TFLJZp;

double *RXp,*RYp,*RZp,*Rp;

double *TRXp,*TRYp,*TRZp ;

double *BpX,*BpY,*BpZ;

double *TBXp,*TBYp,*TBZp;

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

double CIon ;

double ProtonCoat,Proton;

double TBForce1X,TBForce1Y,TBForce1Z;

double TBForcepX,TBForcepY,TBForcepZ;

double TBForceX,TBForceY,TBForceZ;

double TeTBForce1X,TeTBForce1Y,TeTBForce1Z;

double TeTBForcepX,TeTBForcepY,TeTBForcepZ;
```

```
double TeTBForceX,TeTBForceY,TeTBForceZ;

double IX1[4],IY1[4],IZ1[4];

double IX[4],IY[4],IZ[4];

double IXX1,IYY1,IZZ1 ;

double IXX,IYY,IZZ ;

double IonX,IonY,IonZ;

double TIonX,TIonY,TIonZ;

double t,MI;

double LengthX,LengthY,LengthZ;

double Zeus;

long N1X,N1Y,N1Z;

double dx,dy,dz;

long bi,bj,bk;

long step;

long Particle,NParticle;

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

void F_BaseIonForce()

{

    //Zeus = -1.00e-10; GOOD
    //Zeus = -5.00e-12;100eV
    Zeus = -4.00e-12;
    //Zeus = 0.00;

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

TIonX = F_CheckPosition(IonX,LengthX);
TIonY = F_CheckPosition(IonY,LengthY);
//TIonZ = F_CheckPosition(IonZ,LengthZ);
TIonZ = IonZ;

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

for( bi = 0 ; bi < N1X ; bi++)
{
 for( bj = 0 ; bj < N1Y ; bj++)
 {
  for( bk = 0 ; bk < N1Z ; bk++)
  {

     TRX1[bi][bj][bk] = f_TPosition(TIonX,B1X[bi][bj][bk]);
     TRY1[bi][bj][bk] = f_TPosition(TIonY,B1Y[bi][bj][bk]);
     TRZ1[bi][bj][bk] = f_TPosition(TIonZ,B1Z[bi][bj][bk]);

     TFCouX1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRX1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouY1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRY1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouZ1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRZ1[bi][bj][bk],TRX1[bi][bj
```

```
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);

     TFLJX1[bi][bj][bk] = f_LJF(TRX1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJY1[bi][bj][bk] = f_LJF(TRY1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJZ1[bi][bj][bk] = f_LJF(TRZ1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);

     TFX1[bi][bj][bk] = TFCouX1[bi][bj][bk] + TFLJX1[bi][bj][bk] ;
     TFY1[bi][bj][bk] = TFCouY1[bi][bj][bk] + TFLJY1[bi][bj][bk] ;
     TFZ1[bi][bj][bk] = TFCouZ1[bi][bj][bk] + TFLJZ1[bi][bj][bk] ;

  }
 }
}

TBForce1X = f_TForce(TFX1,N1X,N1Y,N1Z);
TBForce1Y = f_TForce(TFY1,N1X,N1Y,N1Z);
TBForce1Z = f_TForce(TFZ1,N1X,N1Y,N1Z);
//-------------------------------------------------------------------------------------
--------------------------

for( bi = 0 ; bi < NParticle ; bi++)

  {
     TRXp[bi] = f_TPosition(TIonX,BpX[bi]);
     TRYp[bi] = f_TPosition(TIonY,BpY[bi]);
     TRZp[bi] = f_TPosition(TIonZ,BpZ[bi]);

     TFCouXp[bi] = f_CoulombF(CQp[bi],CIon,TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouYp[bi] = f_CoulombF(CQp[bi],CIon,TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouZp[bi] = f_CoulombF(CQp[bi],CIon,TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);

     TFLJXp[bi] = f_LJF(TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJYp[bi] = f_LJF(TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJZp[bi] = f_LJF(TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);

     TFXp[bi] = TFCouXp[bi] + TFLJXp[bi] ;
     TFYp[bi] = TFCouYp[bi] + TFLJYp[bi] ;
     TFZp[bi] = TFCouZp[bi] + TFLJZp[bi] ;

    }

TBForcepX = f_ITForce(TFXp,NParticle);
TBForcepY = f_ITForce(TFYp,NParticle);
TBForcepZ = f_ITForce(TFZp,NParticle);


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

//exit(0);

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

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

TBForceX = TBForce1X + TBForcepX ;
TBForceY = TBForce1Y + TBForcepY ;
TBForceZ = TBForce1Z + TBForcepZ ;

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

```
--------------------------

//exit(0);

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

/*-------------------------------------------------------------------------------------*/
/*Runge-Kutta Method */
/*-------------------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------*/
/*ion 1
/*-------------------------------------------------------------------------------------*/

        IX1[0] = (t/MI)*(TBForceX + Zeus*IXX1) ;

        IX[0] =  t*(IXX1+IX1[0]);

        TIonX = IonX + IX[0];

/*-------------------------------------------------------------------------------- */

        IY1[0] = (t/MI)*(TBForceY + Zeus*IYY1);

        IY[0] =  t*(IYY1+IY1[0]);

        TIonY = IonY + IY[0];


/*-------------------------------------------------------------------------------- */

        IZ1[0] = (t/MI)*(TBForceZ + Zeus*IZZ1);

        IZ[0] =  t*(IZZ1+IZ1[0]);

        TIonZ = IonZ + IZ[0];

/*-------------------------------------------------------------------------------- */
/*                                                                                  */
/*-------------------------------------------------------------------------------- */

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

TIonX = F_CheckPosition(TIonX,LengthX);
TIonY = F_CheckPosition(TIonY,LengthY);
//TIonZ = F_CheckPosition(IonZ,LengthZ);


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

for( bi = 0 ; bi < N1X ; bi++)
{
 for( bj = 0 ; bj < N1Y ; bj++)
 {
  for( bk = 0 ; bk < N1Z ; bk++)
  {

     TRX1[bi][bj][bk] = f_TPosition(TIonX,B1X[bi][bj][bk]);
     TRY1[bi][bj][bk] = f_TPosition(TIonY,B1Y[bi][bj][bk]);
     TRZ1[bi][bj][bk] = f_TPosition(TIonZ,B1Z[bi][bj][bk]);

     TFCouX1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRX1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouY1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRY1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
```

```
    TFCouZ1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRZ1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);

    TFLJX1[bi][bj][bk] = f_LJF(TRX1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
    TFLJY1[bi][bj][bk] = f_LJF(TRY1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
    TFLJZ1[bi][bj][bk] = f_LJF(TRZ1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);

    TFX1[bi][bj][bk] = TFCouX1[bi][bj][bk] + TFLJX1[bi][bj][bk] ;
    TFY1[bi][bj][bk] = TFCouY1[bi][bj][bk] + TFLJY1[bi][bj][bk] ;
    TFZ1[bi][bj][bk] = TFCouZ1[bi][bj][bk] + TFLJZ1[bi][bj][bk] ;

  }
 }
}


TeTBForce1X = f_TForce(TFX1,N1X,N1Y,N1Z);
TeTBForce1Y = f_TForce(TFY1,N1X,N1Y,N1Z);
TeTBForce1Z = f_TForce(TFZ1,N1X,N1Y,N1Z);

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

for( bi = 0 ; bi < NParticle ; bi++)

  {
    TRXp[bi] = f_TPosition(TIonX,BpX[bi]);
    TRYp[bi] = f_TPosition(TIonY,BpY[bi]);
    TRZp[bi] = f_TPosition(TIonZ,BpZ[bi]);

    TFCouXp[bi] = f_CoulombF(CQp[bi],CIon,TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
    TFCouYp[bi] = f_CoulombF(CQp[bi],CIon,TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
    TFCouZp[bi] = f_CoulombF(CQp[bi],CIon,TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);

    TFLJXp[bi] = f_LJF(TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
    TFLJYp[bi] = f_LJF(TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
    TFLJZp[bi] = f_LJF(TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);

    TFXp[bi] = TFCouXp[bi] + TFLJXp[bi] ;
    TFYp[bi] = TFCouYp[bi] + TFLJYp[bi] ;
    TFZp[bi] = TFCouZp[bi] + TFLJZp[bi] ;

   }

TeTBForcepX = f_ITForce(TFXp,NParticle);
TeTBForcepY = f_ITForce(TFYp,NParticle);
TeTBForcepZ = f_ITForce(TFZp,NParticle);

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

TeTBForceX = TeTBForce1X + TeTBForcepX;
TeTBForceY = TeTBForce1Y + TeTBForcepY;
TeTBForceZ = TeTBForce1Z + TeTBForcepZ;

//-------------------------------------------------------------------------------------
--------------------------
/*-------------------------------------------------------------------------------*/
/*ion 2
/*-------------------------------------------------------------------------------*/
```

```
        IX1[1] = (t/MI)*(TeTBForceX + Zeus*(IXX1+IX1[0]));

        IX[1] =  t*(IXX1+IX1[1]/2.00);

        TIonX = IonX + IX[1]/2.00;

/*------------------------------------------------------------------------------ */

        IY1[1] = (t/MI)*(TeTBForceY + Zeus*(IYY1+IY1[0]));

        IY[1] =  t*(IYY1+IY1[1]/2.00);

        TIonY = IonY + IY[1]/2.00;


/*------------------------------------------------------------------------------ */

        IZ1[1] = (t/MI)*(TeTBForceZ + Zeus*(IZZ1+IZ1[0]));

        IZ[1] =  t*(IZZ1+IZ1[1]/2.00);

        TIonZ = IonZ + IZ[1]/2.00;

/*------------------------------------------------------------------------------ */
/*                                                                               */
/*------------------------------------------------------------------------------ */

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

TIonX = F_CheckPosition(TIonX,LengthX);
TIonY = F_CheckPosition(TIonY,LengthY);
//TIonZ = F_CheckPosition(IonZ,LengthZ);


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

for( bi = 0 ; bi < N1X ; bi++)
{
 for( bj = 0 ; bj < N1Y ; bj++)
 {
  for( bk = 0 ; bk < N1Z ; bk++)
  {

     TRX1[bi][bj][bk] = f_TPosition(TIonX,B1X[bi][bj][bk]);
     TRY1[bi][bj][bk] = f_TPosition(TIonY,B1Y[bi][bj][bk]);
     TRZ1[bi][bj][bk] = f_TPosition(TIonZ,B1Z[bi][bj][bk]);

     TFCouX1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRX1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouY1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRY1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouZ1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRZ1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);

     TFLJX1[bi][bj][bk] = f_LJF(TRX1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJY1[bi][bj][bk] = f_LJF(TRY1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJZ1[bi][bj][bk] = f_LJF(TRZ1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);


     TFX1[bi][bj][bk] = TFCouX1[bi][bj][bk] + TFLJX1[bi][bj][bk] ;
     TFY1[bi][bj][bk] = TFCouY1[bi][bj][bk] + TFLJY1[bi][bj][bk] ;
     TFZ1[bi][bj][bk] = TFCouZ1[bi][bj][bk] + TFLJZ1[bi][bj][bk] ;
```

```
  }
 }
}

TeTBForce1X = f_TForce(TFX1,N1X,N1Y,N1Z);
TeTBForce1Y = f_TForce(TFY1,N1X,N1Y,N1Z);
TeTBForce1Z = f_TForce(TFZ1,N1X,N1Y,N1Z);

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

for( bi = 0 ; bi < NParticle ; bi++)

  {
     TRXp[bi] = f_TPosition(TIonX,BpX[bi]);
     TRYp[bi] = f_TPosition(TIonY,BpY[bi]);
     TRZp[bi] = f_TPosition(TIonZ,BpZ[bi]);

     TFCouXp[bi] = f_CoulombF(CQp[bi],CIon,TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouYp[bi] = f_CoulombF(CQp[bi],CIon,TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouZp[bi] = f_CoulombF(CQp[bi],CIon,TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);

     TFLJXp[bi] = f_LJF(TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJYp[bi] = f_LJF(TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJZp[bi] = f_LJF(TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);

     TFXp[bi] = TFCouXp[bi] + TFLJXp[bi] ;
     TFYp[bi] = TFCouYp[bi] + TFLJYp[bi] ;
     TFZp[bi] = TFCouZp[bi] + TFLJZp[bi] ;

    }

TeTBForcepX = f_ITForce(TFXp,NParticle);
TeTBForcepY = f_ITForce(TFYp,NParticle);
TeTBForcepZ = f_ITForce(TFZp,NParticle);

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

TeTBForceX = TeTBForce1X + TeTBForcepX;
TeTBForceY = TeTBForce1Y + TeTBForcepY;
TeTBForceZ = TeTBForce1Z + TeTBForcepZ;

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

//-------------------------------------------------------------------------------------
--------------------------
/*-------------------------------------------------------------------------------*/
/*ion 3
/*-------------------------------------------------------------------------------*/

        IX1[2] = (t/MI)*(TeTBForceX + Zeus*(IXX1+(IX1[1]/2.00)));

        IX[2] =  t*(IXX1+IX1[2]/2.00);

        TIonX = IonX + IX[2]/2.00;

/*-------------------------------------------------------------------------------- */

        IY1[2] = (t/MI)*(TeTBForceY + Zeus*(IYY1+(IY1[1]/2.00)));
```

```
        IY[2] =  t*(IYY1+IY1[2]/2.00);

        TIonY = IonY + IY[2]/2.00;

/*------------------------------------------------------------------------------ */

        IZ1[2] = (t/MI)*(TeTBForceZ + Zeus*(IZZ1+(IZ1[1]/2.00)));

        IZ[2] =  t*(IZZ1+IZ1[2]/2.00);

        TIonZ = IonZ + IZ[2]/2.00;

/*------------------------------------------------------------------------------ */
/*                                                                               */
/*------------------------------------------------------------------------------ */

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

TIonX = F_CheckPosition(TIonX,LengthX);
TIonY = F_CheckPosition(TIonY,LengthY);
//TIonZ = F_CheckPosition(IonZ,LengthZ);


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

for( bi = 0 ; bi < N1X ; bi++)
{
 for( bj = 0 ; bj < N1Y ; bj++)
 {
  for( bk = 0 ; bk < N1Z ; bk++)
  {

     TRX1[bi][bj][bk] = f_TPosition(TIonX,B1X[bi][bj][bk]);
     TRY1[bi][bj][bk] = f_TPosition(TIonY,B1Y[bi][bj][bk]);
     TRZ1[bi][bj][bk] = f_TPosition(TIonZ,B1Z[bi][bj][bk]);

     TFCouX1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRX1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouY1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRY1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);
     TFCouZ1[bi][bj][bk] = f_CoulombF(CQ1[bi][bj][bk],CIon,TRZ1[bi][bj][bk],TRX1[bi][bj
][bk],TRY1[bi][bj][bk],TRZ1[bi][bj][bk],ProtonCoat,Proton);

     TFLJX1[bi][bj][bk] = f_LJF(TRX1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJY1[bi][bj][bk] = f_LJF(TRY1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);
     TFLJZ1[bi][bj][bk] = f_LJF(TRZ1[bi][bj][bk],TRX1[bi][bj][bk],TRY1[bi][bj][bk],TRZ1
[bi][bj][bk]);

     TFX1[bi][bj][bk] = TFCouX1[bi][bj][bk] + TFLJX1[bi][bj][bk] ;
     TFY1[bi][bj][bk] = TFCouY1[bi][bj][bk] + TFLJY1[bi][bj][bk] ;
     TFZ1[bi][bj][bk] = TFCouZ1[bi][bj][bk] + TFLJZ1[bi][bj][bk] ;

  }
 }
}


TeTBForce1X = f_TForce(TFX1,N1X,N1Y,N1Z);
TeTBForce1Y = f_TForce(TFY1,N1X,N1Y,N1Z);
TeTBForce1Z = f_TForce(TFZ1,N1X,N1Y,N1Z);

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

```
for( bi = 0 ; bi < NParticle ; bi++)

  {
     TRXp[bi] = f_TPosition(TIonX,BpX[bi]);
     TRYp[bi] = f_TPosition(TIonY,BpY[bi]);
     TRZp[bi] = f_TPosition(TIonZ,BpZ[bi]);

     TFCouXp[bi] = f_CoulombF(CQp[bi],CIon,TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouYp[bi] = f_CoulombF(CQp[bi],CIon,TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);
     TFCouZp[bi] = f_CoulombF(CQp[bi],CIon,TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi],ProtonCo
at,Proton);

     TFLJXp[bi] = f_LJF(TRXp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJYp[bi] = f_LJF(TRYp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);
     TFLJZp[bi] = f_LJF(TRZp[bi],TRXp[bi],TRYp[bi],TRZp[bi]);

     TFXp[bi] = TFCouXp[bi] + TFLJXp[bi] ;
     TFYp[bi] = TFCouYp[bi] + TFLJYp[bi] ;
     TFZp[bi] = TFCouZp[bi] + TFLJZp[bi] ;

     //printf("Position %e,%e,%e,%e,%e,%e,\n",BpX[bi],BpY[bi],BpZ[bi],TRXp[bi],TRYp[bi]
,TRZp[bi]);

    }


TeTBForcepX = f_ITForce(TFXp,NParticle);
TeTBForcepY = f_ITForce(TFYp,NParticle);
TeTBForcepZ = f_ITForce(TFZp,NParticle);

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

TeTBForceX = TeTBForce1X + TeTBForcepX;
TeTBForceY = TeTBForce1Y + TeTBForcepY;
TeTBForceZ = TeTBForce1Z + TeTBForcepZ;

//printf("FORCEX %e,%e,%e,\n",TeTBForceX,TeTBForce1X,TeTBForcepX);
//printf("FORCEY %e,%e,%e,\n",TeTBForceY,TeTBForce1Y,TeTBForcepY);
//printf("FORCEZ %e,%e,%e,\n",TeTBForceZ,TeTBForce1Z,TeTBForcepZ);

//-------------------------------------------------------------------------------------
--------------------------
/*-------------------------------------------------------------------------------*/
/*ion 4
/*-------------------------------------------------------------------------------*/


        IX1[3] = (t/MI)*(TeTBForceX + Zeus*(IXX1+(IX1[2]/2.00)));

        IX[3] =  t*(IXX1+IX1[3]);

/*-------------------------------------------------------------------------------- */

        IY1[3] = (t/MI)*(TeTBForceY + Zeus*(IYY1+(IX1[2]/2.00)));

        IY[3] =  t*(IYY1+IY1[3]);

/*-------------------------------------------------------------------------------- */

        IZ1[3] = (t/MI)*(TeTBForceZ + Zeus*(IZZ1+(IX1[2]/2.00)));

        IZ[3] =  t*(IZZ1+IZ1[3]);

/*-------------------------------------------------------------------------------- */
```

```
/*------------------------------------------------------------------------------ */
        IXX1 = IXX1+(1.00/6.00)*(IX1[0]+2.00*IX1[1]+2.00*IX1[2]+IX1[3]);
        IXX =  (1.00/6.00)*(IX[0]+2.00*IX[1]+2.00*IX[2]+IX[3]);
        IonX = IonX + IXX;

        IYY1 = IYY1+(1.00/6.00)*(IY1[0]+2.00*IY1[1]+2.00*IY1[2]+IY1[3]);
        IYY =  (1.00/6.00)*(IY[0]+2.00*IY[1]+2.00*IY[2]+IY[3]);
        IonY = IonY + IYY;
        IZZ1 = IZZ1+(1.00/6.00)*(IZ1[0]+2.00*IZ1[1]+2.00*IZ1[2]+IZ1[3]);
        IZZ =  (1.00/6.00)*(IZ[0]+2.00*IZ[1]+2.00*IZ[2]+IZ[3]);
        IonZ = IonZ + IZZ;
        printf("%d,%e,%e,%e,%e,%e,%e,%e,\n",NParticle,t*step,IonX,IonY,IonZ,IXX1,IYY1,I
ZZ1);

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>

double IXX1,IYY1,IZZ1 ;

double IXX,IYY,IZZ ;

double IonX,IonY,IonZ;

double t;

char C[40];

int  decimal, sign;

long aoh;

long step;
//-------------------------------------------------------------------------------------
--//

void F_Print()
{
    long pti,ptj,ptk;

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

FILE *fData;


    //C = _fcvt((float)aoh,0,&decimal,&sign);

    sprintf(C,"%d",aoh);

    strcat(C,"Data.csv");

    //printf("%s",C);

    //exit(0);

    fData = fopen(C,"a+t");

    if(fData == NULL)

    {
    printf("ERRorrrr-Phi Particle.csv !\n");

    exit(1);

    }

    else

    {

    fprintf(fData,"%e,%e,%e,%e,%e,%e,%e,\n",t*step,IonX,IonY,IonZ,IXX1,IYY1,IZZ1);

    }

    fclose(fData);


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

```
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

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

double IonX,IonY,IonZ;

double *BpX,*BpY,*BpZ;

long Particle,NParticle;


void F_Particle()

{


    BpX[NParticle] = IonX ;
    BpY[NParticle] = IonY ;
    BpZ[NParticle] = IonZ ;


    NParticle = NParticle + 1;

return;

}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

double ***Q1,***CQ1 ;

double ***FX1,***FY1,***FZ1;

double ***TFX1,***TFY1,***TFZ1;

double ***FCouX1,***FCouY1,***FCouZ1;

double ***FLJX1,***FLJY1,***FLJZ1;

double ***TFCouX1,***TFCouY1,***TFCouZ1;

double ***TFLJX1,***TFLJY1,***TFLJZ1;

double ***RX1,***RY1,***RZ1,***R1;

double ***TRX1,***TRY1,***TRZ1 ;

double ***B1X,***B1Y,***B1Z;

double ***TBX1,***TBY1,***TBZ1;

//--------------------------
double *Qp,*CQp ;

double *FXp,*FYp,*FZp;

double *TFXp,*TFYp,*TFZp;

double *FCouXp,*FCouYp,*FCouZp;

double *FLJXp,*FLJYp,*FLJZp;

double *TFCouXp,*TFCouYp,*TFCouZp;

double *TFLJXp,*TFLJYp,*TFLJZp;

double *RXp,*RYp,*RZp,*Rp;

double *TRXp,*TRYp,*TRZp ;

double *BpX,*BpY,*BpZ;

double *TBXp,*TBYp,*TBZp;
//--------------------------

long N1X,N1Y,N1Z;

long Particle;

void F_Free()

{

long fi,fj,fk,ff;

//-------------------------------------------------------------------------------------
----
    for ( fi = 0 ; fi <N1X ; fi++)
    for ( fj = 0 ; fj <N1Y ; fj++)
    {
    {
        free(Q1[fi][fj]);
```

```
    free(CQ1[fi][fj]) ;

    free(FX1[fi][fj]) ;
    free(FY1[fi][fj]) ;
    free(FZ1[fi][fj]) ;

    free(FCouX1[fi][fj]) ;
    free(FCouY1[fi][fj]) ;
    free(FCouZ1[fi][fj]) ;

    free(FLJX1[fi][fj]) ;
    free(FLJY1[fi][fj]) ;
    free(FLJZ1[fi][fj]) ;

    free(TFX1[fi][fj]) ;
    free(TFY1[fi][fj]) ;
    free(TFZ1[fi][fj]) ;

    free(TFCouX1[fi][fj]) ;
    free(TFCouY1[fi][fj]) ;
    free(TFCouZ1[fi][fj]) ;

    free(TFLJX1[fi][fj]) ;
    free(TFLJY1[fi][fj]) ;
    free(TFLJZ1[fi][fj]) ;

    free(RX1[fi][fj]) ;
    free(RY1[fi][fj]) ;
    free(RZ1[fi][fj]) ;
    free(R1[fi][fj]) ;

    free(TRX1[fi][fj]) ;
    free(TRY1[fi][fj]) ;
    free(TRZ1[fi][fj]) ;

    free(B1X[fi][fj]) ;
    free(B1Y[fi][fj]) ;
    free(B1Z[fi][fj]) ;

    free(TBX1[fi][fj]);
    free(TBY1[fi][fj]);
    free(TBZ1[fi][fj]);

}
}

for ( fi = 0 ; fi <N1X ; fi++)
{
    free(Q1[fi]);
    free(CQ1[fi]);

    free(FX1[fi]);
    free(FY1[fi]);
    free(FZ1[fi]);

    free(FCouX1[fi]);
    free(FCouY1[fi]);
    free(FCouZ1[fi]);

    free(FLJX1[fi]);
    free(FLJY1[fi]);
    free(FLJZ1[fi]);

    free(TFX1[fi]);
    free(TFY1[fi]);
    free(TFZ1[fi]);

    free(TFCouX1[fi]);
```

```
        free(TFCouY1[fi]);
        free(TFCouZ1[fi]);

        free(TFLJX1[fi]);
        free(TFLJY1[fi]);
        free(TFLJZ1[fi]);

        free(RX1[fi]);
        free(RY1[fi]);
        free(RZ1[fi]);
        free(R1[fi]);

        free(TRX1[fi]);
        free(TRY1[fi]);
        free(TRZ1[fi]);

        free(B1X[fi]);
        free(B1Y[fi]);
        free(B1Z[fi]);

        free(TBX1[fi]);
        free(TBY1[fi]);
        free(TBZ1[fi]);

    }

free(Q1);
free(CQ1);

free(FX1);
free(FY1);
free(FZ1);

free(FCouX1);
free(FCouY1);
free(FCouZ1);

free(FLJX1);
free(FLJY1);
free(FLJZ1);

free(TFX1);
free(TFY1);
free(TFZ1);

free(TFCouX1);
free(TFCouY1);
free(TFCouZ1);

free(TFLJX1);
free(TFLJY1);
free(TFLJZ1);


free(RX1);
free(RY1);
free(RZ1);
free(R1);

free(TRX1);
free(TRY1);
free(TRZ1);

free(B1X);
free(B1Y);
free(B1Z);

free(TBX1);
```

```
free(TBY1);
free(TBZ1);

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

free(Qp);
free(CQp);

free(FXp);
free(FYp);
free(FZp);

free(FCouXp);
free(FCouYp);
free(FCouZp);

free(FLJXp);
free(FLJYp);
free(FLJZp);

free(TFXp);
free(TFYp);
free(TFZp);

free(TFCouXp);
free(TFCouYp);
free(TFCouZp);

free(TFLJXp);
free(TFLJYp);
free(TFLJZp);


free(RXp);
free(RYp);
free(RZp);
free(Rp);

free(TRXp);
free(TRYp);
free(TRZp);

free(BpX);
free(BpY);
free(BpZ);

free(TBXp);
free(TBYp);
free(TBZp);

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


return;

}
```

VITAE

Family name:	PIROONPAN
First name:	THANANCHAI
Gender:	Male
Date of birth:	1975-08-05
Place of birth (city, country):	Chumphon, Thailand
Nationality:	Thai

Address:	33/2 Moo 9 Tambon Takdad Muang
City / Province or state:	Chumphon
Postal code:	86000
Country:	Thailand

UN IVERSITY EDUCATION

from June 1997 to March 2000
Bachelor degree of Engineering
Civil Engineering
King Mongkut University of Technology Thonburi, Bangkok, Thailand

---

from June 2001 to March 2002
Diploma of Nuclear Technology
Nuclear Technology
Chulalongkorn University, Bangkok, Thailand

---

from June 2002 to March 2005
Master degree of Engineering
Nuclear Technology
Chulalongkorn University, Bangkok, Thailand
