Appendices
Appendix A

Program A.1:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(A.1) ***** *                    **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=100,n_f=1000000,n_s=1000000;
    double X[n],Y[n];
    double alp,b,ste=0.5,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    cout<<"Scattering angle(thitta) as a function of impact parameter(b):" <<endl;
    for(long int i=1; i<=n; i++)
      {
        X[i-1]=i*ste;
        b=X[i-1];
        ste_f=sqrt(rou_0-b)/n_f;
        rou_m=pro1(b,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b,ste_s);
        ter_s=pro3(b,alp,ste_s,rou_m);
        Y[i-1]=2*b*(ter_f-ter_s);
      }
    for(long int j=1; j<=n; j++)
      if((j-1)%5==0)
        cout<<X[j-1] <<"     " <<Y[j-1] <<endl;
    getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
    return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
    return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
```

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
  {
    long int i01=1000;
    double rou_i01=0.5,rou_f01,err01=0.5e-6;
    while(i01!=0)
      {
        rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
          if(fabs(rou_f01-rou_i01)<err01)
            i01=0;
              else
                rou_i01=rou_f01;
      }
    return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
    long int n02=1000000;
    double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
      t02=shi02;
      ter_f02=2*t02*fun1(b02,b02+t02*t02);
      for(long int i02=1; i02<=(n02/2-1); i02++)
        {
          t02=shi02+2*i02*ste_f02;
          sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
        }
      for(long int j02=1; j02<=(n02/2); j02++)
        {
          t02=shi02+(2*j02-1)*ste_f02;
          sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
        }
      t02=shi02+n02*ste_f02;
      ter_l02=2*t02*fun1(b02,b02+t02*t02);
    return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
    long int n03=1000000;
    double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
      t03=shi03;
      ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
      for(long int i03=1; i03<=(n03/2-1); i03++)
        {
          t03=shi03+2*i03*ste_s03;
          sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      for(long int j03=1; j03<=(n03/2); j03++)
```

```
        {
          t03=shi03+(2*j03-1)*ste_s03;
          sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      t03=shi03+n03*ste_s03;
      ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
    return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Program A.2:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                    * *****  Program(A.2) ***** *                    **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double b04,double xj04,double xi04);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=100,n_f=1000000,n_s=1000000;
    double X1[n],Y1[n],X2[n],Y2[n];
    double alp,b_f,b_s,ste=0.5,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s,mul,sum=0.0;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    for(long int i=1; i<=n; i++)
      {
        X1[i-1]=i*ste;
        b_f=X1[i-1];
        ste_f=sqrt(rou_0-b_f)/n_f;
        rou_m=pro1(b_f,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b_f,ste_s);
        ter_s=pro3(b_f,alp,ste_s,rou_m);
        Y1[i-1]=2*b_f*(ter_f-ter_s);
      }
    for(long int j=1; j<=n; j++)
      {
        X2[j-1]=X1[j-1];
        b_s=X2[j-1];
        sum=0.0;
```

```
        for(long int k=1; k<=n; k++)
          {
            mul=1.0;
            for(long int l=1; l<=n; l++)
              if(l!=k)
                 mul=mul*pro4(b_s,X1[l-1],X1[k-1]);
            sum=sum+mul*Y1[k-1];
          }
        Y2[j-1]=sum;
      }
    cout<<"Scattering angle(thitta) as a function of impact parameter(b):" <<endl;
    for(long int m=1; m<=n; m++)
      if((m-1-2)%5==0)
        cout<<X2[m-1] <<"     " <<Y2[m-1] <<endl;
    getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
    return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
   return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
  {
   long int i01=1000;
   double rou_i01=0.5,rou_f01,err01=0.5e-6;
   while(i01!=0)
     {
      rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
       if(fabs(rou_f01-rou_i01)<err01)
         i01=0;
           else
             rou_i01=rou_f01;
     }
   return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
    long int n02=1000000;
    double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
      t02=shi02;
      ter_f02=2*t02*fun1(b02,b02+t02*t02);
```

```
      for(long int i02=1; i02<=(n02/2-1); i02++)
        {
         t02=shi02+2*i02*ste_f02;
         sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
        }
      for(long int j02=1; j02<=(n02/2); j02++)
        {
         t02=shi02+(2*j02-1)*ste_f02;
         sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
        }
      t02=shi02+n02*ste_f02;
      ter_l02=2*t02*fun1(b02,b02+t02*t02);
    return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
    long int n03=1000000;
    double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
      t03=shi03;
      ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
      for(long int i03=1; i03<=(n03/2-1); i03++)
        {
         t03=shi03+2*i03*ste_s03;
         sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      for(long int j03=1; j03<=(n03/2); j03++)
        {
         t03=shi03+(2*j03-1)*ste_s03;
         sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      t03=shi03+n03*ste_s03;
      ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
    return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */


double pro4(double b04,double xj04,double xi04)
  {
    return (b04-xj04)/(xi04-xj04);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Appendix B

Program B.1:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(B.1) ***** *                  **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
```

```
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double yj_p04,double yj_m04,double ste04);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=1000,n_f=1000000,n_s=1000000;
    double X1[n],Y1[n],X2[n-2],Y2[n-2],X3[n/10],Y3[n/10];
    double alp,b,ste=0.05,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    cout<<"Differential scattering cross section(sigma) as a function of scattering";
    cout<<" " <<"angle(thitta):" <<endl;
    for(long int i=1; i<=n; i++)
      {
        X1[i-1]=i*ste;
        b=X1[i-1];
        ste_f=sqrt(rou_0-b)/n_f;
        rou_m=pro1(b,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b,ste_s);
        ter_s=pro3(b,alp,ste_s,rou_m);
        Y1[i-1]=2*b*(ter_f-ter_s);
      }
    for(long int j=1; j<=n-2; j++)
      {
        Y2[(j-1)]=Y1[j];
        X2[(j-1)]=fabs(pow(pro4(Y1[j+1],Y1[j-1],ste),-1.0));
      }
    for(long int k=1; k<=n-2; k++)
      if((k-1)%10==0)
        {
          Y3[(k-1)/10]=Y2[k-1];
          X3[(k-1)/10]=X2[k-1];
        }
    for(long int l=1; l<=n/10; l++)
      if((l-1)%5==0)
        cout<<Y3[l-1] <<"     " <<X3[l-1] <<endl;
    getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
    return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
    return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
```

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
  {
    long int i01=1000;
    double rou_i01=0.5,rou_f01,err01=0.5e-6;
    while(i01!=0)
      {
        rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
          if(fabs(rou_f01-rou_i01)<err01)
            i01=0;
              else
                rou_i01=rou_f01;
      }
    return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
     long int n02=1000000;
     double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
       t02=shi02;
       ter_f02=2*t02*fun1(b02,b02+t02*t02);
       for(long int i02=1; i02<=(n02/2-1); i02++)
         {
          t02=shi02+2*i02*ste_f02;
          sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
         }
       for(long int j02=1; j02<=(n02/2); j02++)
         {
          t02=shi02+(2*j02-1)*ste_f02;
          sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
         }
       t02=shi02+n02*ste_f02;
       ter_l02=2*t02*fun1(b02,b02+t02*t02);
     return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
     long int n03=1000000;
     double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
       t03=shi03;
       ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
       for(long int i03=1; i03<=(n03/2-1); i03++)
         {
          t03=shi03+2*i03*ste_s03;
          sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
         }
       for(long int j03=1; j03<=(n03/2); j03++)
         {
          t03=shi03+(2*j03-1)*ste_s03;
          sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
         }
       t03=shi03+n03*ste_s03;
       ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
     return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */

double pro4(double yj_p04,double yj_m04,double ste04)
  {
    return (yj_p04-yj_m04)/(2*ste04);
  }

/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Program B.2:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(B.2) ***** *                  **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double yj_p04,double yj_m04,double ste04);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=1000,n_f=1000000,n_s=1000000;
    double X1[n],Y1[n],X2[n-2],Y2[n-2],X3[n/10],Y3[n/10],X4[n/100],Y4[n/100];
    double alp,b,ste=0.05,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    for(long int i=1; i<=n; i++)
      {
        X1[i-1]=i*ste;
        b=X1[i-1];
        ste_f=sqrt(rou_0-b)/n_f;
        rou_m=pro1(b,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b,ste_s);
        ter_s=pro3(b,alp,ste_s,rou_m);
        Y1[i-1]=2*b*(ter_f-ter_s);
      }
    for(long int j=1; j<=n-2; j++)
      {
        Y2[(j-1)]=Y1[j];
        X2[(j-1)]=fabs(pow(pro4(Y1[j+1],Y1[j-1],ste),-1.0));
      }
    for(long int k=1; k<=n-2; k++)
      if((k-1)%10==0)
        {
          Y3[(k-1)/10]=Y2[k-1];
          X3[(k-1)/10]=X2[k-1];
        }
    for(long int l=21; l<=n/50+10; l++)
      {
        Y4[l-21]=Y3[l-1];
        X4[l-21]=X3[l-1];
      }
    cout<<"Differential scattering cross section(sigma) as a function of scattering";
    cout<<" " <<"angle(thitta):" <<endl;
    for(long int m=1; m<=n/100; m++)
      cout<<Y4[m-1] <<"     " <<X4[m-1] <<endl;
    getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
     return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
```

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
   return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
  {
   long int i01=1000;
   double rou_i01=0.5,rou_f01,err01=0.5e-6;
   while(i01!=0)
     {
       rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
         if(fabs(rou_f01-rou_i01)<err01)
           i01=0;
             else
               rou_i01=rou_f01;
     }
   return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
    long int n02=1000000;
    double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
      t02=shi02;
      ter_f02=2*t02*fun1(b02,b02+t02*t02);
      for(long int i02=1; i02<=(n02/2-1); i02++)
        {
         t02=shi02+2*i02*ste_f02;
         sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
        }
      for(long int j02=1; j02<=(n02/2); j02++)
        {
         t02=shi02+(2*j02-1)*ste_f02;
         sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
        }
      t02=shi02+n02*ste_f02;
      ter_l02=2*t02*fun1(b02,b02+t02*t02);
    return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
    long int n03=1000000;
    double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
      t03=shi03;
      ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
      for(long int i03=1; i03<=(n03/2-1); i03++)
```

```
        {
         t03=shi03+2*i03*ste_s03;
         sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      for(long int j03=1; j03<=(n03/2); j03++)
        {
         t03=shi03+(2*j03-1)*ste_s03;
         sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      t03=shi03+n03*ste_s03;
      ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
    return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */

double pro4(double yj_p04,double yj_m04,double ste04)
  {
    return (yj_p04-yj_m04)/(2*ste04);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Program B.3:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(B.3) ***** *                  **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double yj_p04,double yj_m04,double ste04);
double pro5(double thi05,double yj05,double yi05);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=1000,n_f=1000000,n_s=1000000;
    double X1[n],Y1[n],X2[n-2],Y2[n-2],X3[n/10],Y3[n/10],X4[n/100],Y4[n/100],X5[n/100];
    double Y5[n/100];
    double alp,b,ste=0.05,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s,thi,mul,sum=0.0;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    for(long int i=1; i<=n; i++)
      {
        X1[i-1]=i*ste;
        b=X1[i-1];
        ste_f=sqrt(rou_0-b)/n_f;
        rou_m=pro1(b,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b,ste_s);
        ter_s=pro3(b,alp,ste_s,rou_m);
        Y1[i-1]=2*b*(ter_f-ter_s);
      }
    for(long int j=1; j<=n-2; j++)
      {
        Y2[(j-1)]=Y1[j];
        X2[(j-1)]=fabs(pow(pro4(Y1[j+1],Y1[j-1],ste),-1.0));
      }
```

```
   for(long int k=1; k<=n-2; k++)
     if((k-1)%10==0)
       {
         Y3[(k-1)/10]=Y2[k-1];
         X3[(k-1)/10]=X2[k-1];
       }
   for(long int l=21; l<=n/50+10; l++)
     {
       Y4[l-21]=Y3[l-1];
       X4[l-21]=X3[l-1];
     }
   for(long int m=1; m<=n/100; m++)
     {
       Y5[m-1]=Y4[m-1];
       thi=Y5[m-1];
       sum=0.0;
       for(long int u=1; u<=n/100; u++)
         {
           mul=1.0;
           for(long int w=1; w<=n/100; w++)
             if(w!=u)
               mul=mul*pro5(thi,Y4[w-1],Y4[u-1]);
           sum=sum+mul*X4[u-1];
         }
       X5[m-1]=sum;
     }
   cout<<"Differential scattering cross section(sigma) as a function of scattering";
   cout<<" " <<"angle(thitta):" <<endl;
   for(long int v=1; v<=n/100; v++)
     cout<<Y5[v-1] <<"     " <<X5[v-1] <<endl;
   getch();
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
 {
   return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
 {
  return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
 {
   return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
 {
   return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
 {
   double rou5_0=1e+4;
   return log(rou5_0/rou5);
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
 {
  long int i01=1000;
  double rou_i01=0.5,rou_f01,err01=0.5e-6;
  while(i01!=0)
```

```
    {
      rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
        if(fabs(rou_f01-rou_i01)<err01)
          i01=0;
            else
              rou_i01=rou_f01;
    }
   return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
    long int n02=1000000;
    double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
      t02=shi02;
      ter_f02=2*t02*fun1(b02,b02+t02*t02);
      for(long int i02=1; i02<=(n02/2-1); i02++)
        {
          t02=shi02+2*i02*ste_f02;
          sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
        }
      for(long int j02=1; j02<=(n02/2); j02++)
        {
          t02=shi02+(2*j02-1)*ste_f02;
          sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
        }
      t02=shi02+n02*ste_f02;
      ter_l02=2*t02*fun1(b02,b02+t02*t02);
    return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
    long int n03=1000000;
    double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
      t03=shi03;
      ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
      for(long int i03=1; i03<=(n03/2-1); i03++)
        {
          t03=shi03+2*i03*ste_s03;
          sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      for(long int j03=1; j03<=(n03/2); j03++)
        {
          t03=shi03+(2*j03-1)*ste_s03;
          sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      t03=shi03+n03*ste_s03;
      ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
    return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro4(double yj_p04,double yj_m04,double ste04)
  {
    return (yj_p04-yj_m04)/(2*ste04);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro5(double thi05,double yj05,double yi05)
  {
    return (thi05-yj05)/(yi05-yj05);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Appendix C

Program C.1:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(C.1) ***** *                  **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
```

```
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double yj_p04,double yj_m04,double ste04);
double pro5(double thi05,double yj05,double yi05);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n=1000,n_f=1000000,n_s=1000000;
    double X1[n],Y1[n],X2[n-2],Y2[n-2],X3[n/10],Y3[n/10],X4[n/100],Y4[n/100],X5[n/100];
    double Y5[n/100];
    double alp,b,ste_fe=0.05,ste_se,ste_f,ste_s,rou_0=1e+4,rou_m,ter_f,ter_s,thi;
    double thi_i=2.12546,thi_f=2.04422,mul,sum;
    cout<<"Enter ratio between linear charge density and total energy(alpha):" <<endl;
    cin>>alp;
    for(long int i=1; i<=n; i++)
      {
        X1[i-1]=i*ste_fe;
        b=X1[i-1];
        ste_f=sqrt(rou_0-b)/n_f;
        rou_m=pro1(b,alp);
        ste_s=sqrt(rou_0-rou_m)/n_s;
        ter_f=pro2(b,ste_s);
        ter_s=pro3(b,alp,ste_s,rou_m);
        Y1[i-1]=2*b*(ter_f-ter_s);
      }
    for(long int j=1; j<=n-2; j++)
      {
        Y2[(j-1)]=Y1[j];
        X2[(j-1)]=fabs(pow(pro4(Y1[j+1],Y1[j-1],ste_fe),-1.0));
      }
    for(long int k=1; k<=n-2; k++)
      if((k-1)%10==0)
        {
          Y3[(k-1)/10]=Y2[k-1];
          X3[(k-1)/10]=X2[k-1];
        }
    for(long int l=21; l<=n/50+10; l++)
      {
        Y4[l-21]=Y3[l-1];
        X4[l-21]=X3[l-1];
      }
    cout<<"Differential scattering cross section(sigma) as a function of scattering";
    cout<<" " <<"angle(thitta):" <<endl;
    ste_se=(thi_f-thi_i)/(n/100);
    for(long int m=1; m<=n/100; m++)
      {
        Y5[m-1]=thi_i+(m-1)*ste_se;
        thi=Y5[m-1];
        sum=0.0;
        for(long int u=1; u<=n/100; u++)
          {
            mul=1.0;
            for(long int v=1; v<=n/100; v++)
              if(v!=u)
                mul=mul*pro5(thi,Y4[v-1],Y4[u-1]);
            sum=sum+mul*X4[u-1];
          }
        X5[m-1]=sum;
      }
    for(long int w=1; w<=n/100; w++)
      cout<<Y5[w-1] <<"     " <<X5[w-1] <<endl;
    getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
```

```
    return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
   return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro1(double b01,double alp01)
  {
   long int i01=1000;
   double rou_i01=0.5,rou_f01,err01=0.5e-6;
   while(i01!=0)
     {
      rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
        if(fabs(rou_f01-rou_i01)<err01)
          i01=0;
            else
              rou_i01=rou_f01;
     }
   return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
    long int n02=1000000;
    double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
      t02=shi02;
      ter_f02=2*t02*fun1(b02,b02+t02*t02);
      for(long int i02=1; i02<=(n02/2-1); i02++)
        {
         t02=shi02+2*i02*ste_f02;
         sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
        }
      for(long int j02=1; j02<=(n02/2); j02++)
        {
         t02=shi02+(2*j02-1)*ste_f02;
         sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
        }
      t02=shi02+n02*ste_f02;
      ter_l02=2*t02*fun1(b02,b02+t02*t02);
    return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
    long int n03=1000000;
    double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
      t03=shi03;
```

```
      ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
      for(long int i03=1; i03<=(n03/2-1); i03++)
        {
         t03=shi03+2*i03*ste_s03;
         sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      for(long int j03=1; j03<=(n03/2); j03++)
        {
         t03=shi03+(2*j03-1)*ste_s03;
         sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
        }
      t03=shi03+n03*ste_s03;
      ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
    return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro4(double yj_p04,double yj_m04,double ste04)
  {
    return (yj_p04-yj_m04)/(2*ste04);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro5(double thi05,double yj05,double yi05)
  {
    return (thi05-yj05)/(yi05-yj05);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Program C.2:

```
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* ****                  * *****  Program(C.2) ***** *                  **** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
#include <iostream.h>
#include <conio.h>
#include <math.h>
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1);
double fun2(double b2,double rou2,double alp2);
double fun3(double b3,double rou3,double alp3);
double fun4(double b4,double rou4,double alp4);
double fun5(double rou5);
double pro1(double b01,double alp01);
double pro2(double b02,double ste_f02);
double pro3(double b03,double alp03,double ste_s03,double rou_m03);
double pro4(double yj_p04,double yj_m04,double ste04);
double pro5(double thi05,double yj05,double yi05);
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
main()
  {
    long int n_se=1000,n_f=1000000,n_s=1000000;
    double X1[n_se],Y1[n_se],X2[n_se-2],Y2[n_se-2],X3[n_se/10],Y3[n_se/10],X4[n_se/100];
    double Y4[n_se/100],X5[n_se/100],Y5[n_se/100],X6[n_se/100],Y6[n_se/100],Z[n_se/200];
    double alp_fi,alp_fi0=0.2,alp_se,b,ste_fe=0.05,ste_se,ste_th=0.0025,ste_f,ste_s;
    double rou_0=1e+4,rou_m,ter_f,ter_s,thi,thi_i=2.12546,thi_f=2.04422,mul,sum_f,sum_s;
    double min;
    ste_se=(thi_f-thi_i)/(n_se/100);
    for(long int i=1; i<=n_se/100; i++)
      {
        Y6[i-1]=thi_i+(i-1)*ste_se;
        cout<<"Enter differential scattering cross section(sigma) of scattering";
        cout<<" " <<"angle(thitta) equal to:" <<" " <<Y6[i-1] <<endl;
        cin>>X6[i-1];
      }
    for(long int j=1; j<=n_se/200; j++)
      {
        alp_fi=alp_fi0+(j-1)*ste_th;
        for(long int k=1; k<=n_se; k++)
          {
            X1[k-1]=k*ste_fe;
            b=X1[k-1];
            ste_f=sqrt(rou_0-b)/n_f;
            rou_m=pro1(b,alp_fi);
            ste_s=sqrt(rou_0-rou_m)/n_s;
            ter_f=pro2(b,ste_s);
            ter_s=pro3(b,alp_fi,ste_s,rou_m);
            Y1[k-1]=2*b*(ter_f-ter_s);
          }
        for(long int l=1; l<=n_se-2; l++)
          {
            Y2[(l-1)]=Y1[l];
            X2[(l-1)]=fabs(pow(pro4(Y1[l+1],Y1[l-1],ste_fe),-1.0));
          }
        for(long int m=1; m<=n_se-2; m++)
          if((m-1)%10==0)
            {
              Y3[(m-1)/10]=Y2[m-1];
              X3[(m-1)/10]=X2[m-1];
```

```
        }
      for(long int n=21; n<=n_se/50+10; n++)
        {
         Y4[n-21]=Y3[n-1];
         X4[n-21]=X3[n-1];
        }
      for(long int u=1; u<=n_se/100; u++)
        {
          Y5[u-1]=thi_i+(u-1)*ste_se;
          thi=Y5[u-1];
          sum_f=0.0;
          for(long int v=1; v<=n_se/100; v++)
            {
              mul=1.0;
              for(long int w=1; w<=n_se/100; w++)
                if(w!=v)
                mul=mul*pro5(thi,Y4[w-1],Y4[v-1]);
              sum_f=sum_f+mul*X4[v-1];
            }
          X5[u-1]=sum_f;
        }
      sum_s=0.0;
      for(long int a=1; a<=n_se/100; a++)
       sum_s=sum_s+fabs(X5[a-1]-X6[a-1]);
      Z[j-1]=sum_s;
     }
   min=Z[0];
   for(long int c=1; c<=n_se/200; c++)
    if(Z[c-1]<=min)
      {
        min=Z[c-1];
        alp_se=alp_fi0+(c-1)*ste_th;
      }
   cout<<"The ratio between linear charge density and total energy is" <<endl;
   cout<<alp_se <<endl;
   getch();
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun1(double b1,double rou1)
  {
    return pow(rou1*rou1,-1.0)*pow((1.0-b1*b1/(rou1*rou1)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun2(double b2,double rou2,double alp2)
  {
   return pow(rou2*rou2,-1.0)*pow((1.0-b2*b2/(rou2*rou2)-alp2*fun5(rou2)),-0.5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun3(double b3,double rou3,double alp3)
  {
    return 1.0-b3*b3/(rou3*rou3)-alp3*fun5(rou3);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun4(double b4,double rou4,double alp4)
  {
    return (1.0/rou4)*(2*b4*b4/(rou4*rou4)+alp4);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double fun5(double rou5)
  {
    double rou5_0=1e+4;
    return log(rou5_0/rou5);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

```
double pro1(double b01,double alp01)
  {
    long int i01=1000;
    double rou_i01=0.5,rou_f01,err01=0.5e-6;
    while(i01!=0)
      {
        rou_f01=rou_i01-fun3(b01,rou_i01,alp01)/fun4(b01,rou_i01,alp01);
          if(fabs(rou_f01-rou_i01)<err01)
            i01=0;
              else
                rou_i01=rou_f01;
      }
    return rou_f01;
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro2(double b02,double ste_f02)
  {
     long int n02=1000000;
     double sum_eve02=0,sum_odd02=0,ter_f02,ter_l02,shi02=0.5e-6,t02;
       t02=shi02;
       ter_f02=2*t02*fun1(b02,b02+t02*t02);
       for(long int i02=1; i02<=(n02/2-1); i02++)
         {
           t02=shi02+2*i02*ste_f02;
           sum_eve02=sum_eve02+2*t02*fun1(b02,b02+t02*t02);
         }
       for(long int j02=1; j02<=(n02/2); j02++)
         {
           t02=shi02+(2*j02-1)*ste_f02;
           sum_odd02=sum_odd02+2*t02*fun1(b02,b02+t02*t02);
         }
       t02=shi02+n02*ste_f02;
       ter_l02=2*t02*fun1(b02,b02+t02*t02);
     return (ste_f02/3.0)*(ter_f02+2*sum_eve02+4*sum_odd02+ter_l02);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro3(double b03,double alp03,double ste_s03,double rou_m03)
  {
     long int n03=1000000;
     double sum_eve03=0,sum_odd03=0,ter_f03,ter_l03,shi03=0.5e-6,t03;
       t03=shi03;
       ter_f03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
       for(long int i03=1; i03<=(n03/2-1); i03++)
         {
           t03=shi03+2*i03*ste_s03;
           sum_eve03=sum_eve03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
         }
       for(long int j03=1; j03<=(n03/2); j03++)
         {
           t03=shi03+(2*j03-1)*ste_s03;
           sum_odd03=sum_odd03+2*t03*fun2(b03,rou_m03+t03*t03,alp03);
         }
       t03=shi03+n03*ste_s03;
       ter_l03=2*t03*fun2(b03,rou_m03+t03*t03,alp03);
     return (ste_s03/3.0)*(ter_f03+2*sum_eve03+4*sum_odd03+ter_l03);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */

double pro4(double yj_p04,double yj_m04,double ste04)
  {
     return (yj_p04-yj_m04)/(2*ste04);
  }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
double pro5(double thi05,double yj05,double yi05)
  {
     return (thi05-yj05)/(yi05-yj05);
```

```
 }
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
/* **** ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** **** ***** ***** */
```

Appendix D
Simulated Data

Table D.1:

***α*=0.2000**

Impact Parameter(b)	Scattering Angle(*θ*)
0.5	3.08279
3	2.79449
5.5	2.52824
8	2.29458
10.5	2.09596
13	1.92964
15.5	1.79086
18	1.67464
20.5	1.57658

23	1.49312
25.5	1.4214
28	1.3592
30.5	1.30479
33	1.25679
35.5	1.21415
38	1.17599
40.5	1.14163
43	1.11053
45.5	1.08221
48	1.05632

Table D.2:

α=0.2050

Impact Parameter(b)	Scattering Angle(θ)
0.5	3.09017
3	2.83692
5.5	2.59886
8	2.3843
10.5	2.19649
13	2.03486
15.5	1.89676
18	1.77886
20.5	1.67784
23	1.59079
25.5	1.51528
28	1.44929

30.5	**1.39121**
33	**1.33975**
35.5	**1.29385**
38	**1.25266**
40.5	**1.21548**
43	**1.18176**
45.5	**1.15102**
48	**1.12288**

Table D.3:

α=0.2100

Impact Parameter(b)	Scattering Angle(θ)
0.5	**3.09635**
3	**2.8728**
5.5	**2.65982**
8	**2.4638**
10.5	**2.28799**
13	**2.13299**
15.5	**1.99763**
18	**1.87986**
20.5	**1.77734**
23	**1.68786**
25.5	**1.60941**
28	**1.54027**
30.5	**1.47901**
33	**1.42441**

35.5	**1.37549**
38	**1.33142**
40.5	**1.29154**
43	**1.25527**
45.5	**1.22214**
48	**1.19175**

Appendix E
Fitted Data

Table E.1:
α=0.2000

Impact Parameter(b)	Scattering Angle(θ)
1.5	2.96585
4	2.68455
6.5	2.43057
9	2.211
11.5	2.0258
14	1.87111
16.5	1.74193
19	1.63345
21.5	1.54162
24	1.46316
26.5	1.39549
29	1.33659
31.5	1.28489
34	1.23915
36.5	1.19839
39	1.16182
41.5	1.12883
44	1.09889
46.5	1.07158
49	1.04657

Table E.2:

α=0.2050

Impact Parameter(b)	Scattering Angle(θ)
1.5	2.98778
4	2.73929

6.5	**2.50993**
9	**2.30593**
11.5	**2.12881**
14	**1.97698**
16.5	**1.84737**
19	**1.7366**
21.5	**1.64149**
24	**1.55932**
26.5	**1.48784**
29	**1.42519**
31.5	**1.3699**
34	**1.32077**
36.5	**1.27685**
39	**1.23734**
41.5	**1.20161**
44	**1.16913**
46.5	**1.13947**
49	**1.11227**

Table E.3:

α=0.2100

Impact Parameter(b)	Scattering Angle(θ)
1.5	**3.00618**
4	**2.78591**
6.5	**2.57913**

9	**2.39096**
11.5	**2.22353**
14	**2.0766**
16.5	**1.94854**
19	**1.83715**
21.5	**1.74011**
24	**1.65526**
26.5	**1.58073**
29	**1.51489**
31.5	**1.45643**
34	**1.40421**
36.5	**1.35732**
39	**1.315**
41.5	**1.27662**
44	**1.24166**
46.5	**1.20967**
49	**1.18029**

Appendix F
Simulated Data

Table F.1:

α=0.2000

Impact Parameter(b)	Scattering Angle(θ)
3.12982	8.50158
2.83953	8.8274
2.56882	9.7574
2.3296	11.2774
2.12546	13.3689
1.95427	16.0095
1.81143	19.1744
1.69191	22.8506
1.59121	27.0022
1.50562	31.6114
1.43218	36.6686
1.36859	42.1352
1.31303	48.0229
1.26409	54.322
1.22065	60.9899
1.18182	67.9954

1.1469	**75.4214**
1.1153	**83.0332**
1.08657	**91.093**
1.06031	**99.5341**

Table F.2:

α=0.2050

Impact Parameter(b)	Scattering Angle(θ)
3.1313	**9.72347**
2.8767	**10.0076**
2.63553	**10.8221**
2.41686	**12.1555**
2.22473	**13.996**
2.05905	**16.3237**
1.91741	**19.1293**
1.79651	**22.3839**
1.69299	**26.0785**
1.60389	**30.1966**
1.52667	**34.7278**
1.45928	**39.6439**
1.40003	**44.931**
1.34759	**50.5739**
1.30086	**56.5771**
1.25896	**62.9276**
1.22119	**69.5845**
1.18695	**76.561**
1.15576	**83.8636**
1.12722	**91.4719**

Table F.3:

α=0.2100

Impact Parameter(b)	Scattering Angle(θ)
3.13254	11.0533
2.90806	11.3023
2.69289	12.0195
2.49385	13.1952
2.31471	14.8209
2.15643	16.887
2.01805	19.3746
1.89761	22.2742
1.79281	25.5727
1.70139	29.2567
1.6213	33.3236
1.55077	37.7251
1.48833	42.4994
1.43274	47.5791
1.38297	52.999
1.33818	58.7924
1.29766	64.8148
1.26085	71.121
1.22724	77.7707
1.19644	84.6796

Appendix G
Simulated Data

Table G.1:

α=0.2000

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.12546	13.3689
2.08871	13.8531
2.05326	14.36
2.01906	14.889
1.98608	15.4381

1.95427	**16.0095**
1.9236	**16.599**
1.89402	**17.2148**
1.86549	**17.8456**
1.83797	**18.5021**

Table G.2:

α=0.2050

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.22473	**13.996**
2.18953	**14.422**
2.15539	**14.8671**
2.12227	**15.3352**
2.09017	**15.8198**
2.05905	**16.3237**
2.0289	**16.848**
1.99969	**17.3889**
1.97139	**17.9512**
1.94397	**18.5308**

Table G.3:

α=0.2100

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.31471	14.8209
2.28139	15.1993
2.24892	15.5936
2.21727	16.0066
2.18644	16.4387
2.15643	16.887
2.12721	17.3492
2.09879	17.8306
2.07113	18.3314
2.04422	18.8412

Appendix H
Fitted Data

Table H.1:

α=0.2000

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.12546	13.3689
2.08871	13.8531
2.05326	14.36
2.01906	14.889
1.98608	15.4381
1.95427	16.0095
1.9236	16.599
1.89402	17.2148
1.86549	17.8456
1.83797	18.5021

Table H.2:

α=0.2050

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.22473	13.996
2.18953	14.422
2.15539	14.8671
2.12227	15.3352
2.09017	15.8198
2.05905	16.3237
2.0289	16.848
1.99969	17.3889
1.97139	17.9512
1.94397	18.5308

Table H.3:

α=0.2100

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.31471	14.8209

2.28139	**15.1993**
2.24892	**15.5936**
2.21727	**16.0066**
2.18644	**16.4387**
2.15643	**16.887**
2.12721	**17.3492**
2.09879	**17.8306**
2.07113	**18.3314**
2.04422	**18.8412**

Appendix I

Table I.1:

α=0.2000

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.12546	13.3688
2.11734	13.4901
2.10921	13.5944
2.10109	13.6948
2.09296	13.7975
2.08484	13.9049
2.07672	14.017
2.06859	14.1332
2.06047	14.2524
2.05234	14.3737

Table I.2:

α=0.2050

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.12546	13.3688
2.11734	13.4901
2.10921	13.5944
2.10109	13.6948
2.09296	13.7975

2.08484	**13.9049**
2.07672	**14.017**
2.06859	**14.1332**
2.06047	**14.2524**
2.05234	**14.3737**

Table I.3:

α=0.2100

Scattering Angle(θ)	Differential Scattering Cross Section(σ)
2.12546	**17.3779**
2.11734	**17.5125**
2.10921	**17.6499**
2.10109	**17.7903**
2.09296	**17.9336**
2.08484	**18.0796**
2.07672	**18.2281**
2.06859	**18.3785**

2.06047	**18.5305**
2.05234	**18.6844**