Thomas Haschka
Matriculation No.: 0226291

under the Guidance of

Ao. Univ. Prof. Dipl-Ing. Dr. techn.
Helmut Leeb

Summersemester 2006

Student Project:

Numerical Methods of Nuclear Physics

Contents

1 Introduction

The problem to be solved by this student project, is to construct a program that calculates the changes of the scattering phase shifts due to the long range contribution of the potential generated by the polarizeability of the neutron incident on a target nucleus.

We approach the problem by means of the Variable Phase method [4]. This method leads to a non linear differential equation for the phase shifts which describes the change in scattering phase by a term whose size is proportional to the potential. Hence, the method is best suited to evaluate the changes due to small potential contributions. This equation is solved by an adaptive step size Runge Kutta algorithm. Specifically for this program we created a routine to calculate the spherical Bessel and Neumann functions.

The entire program has been written in Fortran 77 in order to achieve compatability with other programs in this field, that may call the program as a subroutine.

2 The Potential caused by Polarization

We start our consideration with a brief sketch of the derivation of the potential contribution generated by the polarizeablility of the neutron. The polarization of the incident particle is caused by the E-field of the nucleus. At present we restrict our sketches to the contribution of the polarizeablity generated by the Coulomb potential of the nucleus, while the corresponding contribution of the electron shells are ignored. The scalar potential generated by the charge of the nucleus is given by

$$\varphi_N(r) = \frac{Ze_0}{4\pi\epsilon_0}\frac{1}{r}. \tag{1}$$

Hence, the electric field strength takes the form

$$\vec{E} = -\frac{d}{dr}(\varphi_N(r))\hat{r} = \frac{Ze_0}{4\pi\epsilon_0}\frac{1}{r^2}\hat{r}. \tag{2}$$

The electric field induces an electric dipole moment of the neutron

$$\vec{D} = 4\pi\epsilon_0\alpha_n\vec{E}, \tag{3}$$

where α_n is the electric polarizability of the neutron. The induced electric dipole moment interacts with the electric field of the nucleus, thus leading to the potential term

$$V_p(r) = -\frac{1}{2}\vec{D}\cdot\vec{E} = -\frac{1}{2}4\pi\epsilon_0\alpha_n\vec{E}^2 = -\frac{1}{2}4\pi\epsilon_0\alpha_n\left(\frac{Ze_0}{4\pi\epsilon_0}\right)^2\frac{1}{r^4}. \tag{4}$$

This term can be cast into the form

$$V_p(r) = -\frac{1}{2}Z^2\alpha_f\hbar c\alpha_n\frac{1}{r^4}, \tag{5}$$

where α_f is the dimensionless fine structure constant.

3 The Variable Phase Method

The effect of the long range part on the scattering phase shifts is best determined by the Variable Phase Method [4]. The potential due to the polarizability vanishes as $\frac{1}{r^4}$ for $r \to \infty$, which is the only contribution outside the nucleus for neutron- nucleus scattering. It should be remarked that the variable phase method allows the evaluation of the wave function for the entire range $0 < r < \infty$. In this project we restrict ourselves to the evaluation of the scattering phase shifts.

Lets briefly sketch the basics of the variable phase method. We start with the radial Schrödinger equation

$$\left[-\frac{\hbar}{2m}\left(\frac{d^2}{dr^2} - \frac{l(l+1)}{r^2}\right) + V(r)\right]\psi_l(r) = E\psi_l(r), \tag{6}$$

which can be written in the form:

$$\left[\frac{d^2}{dr^2} - \frac{l(l+1)}{r^2} + k^2\left(1 - \frac{V(r)}{E}\right)\right]\psi(r) = 0 \tag{7}$$

It is straightforward to show that the logarithmic derivative

$$Z(r) = \frac{d}{dr}ln(\psi(r)) \tag{8}$$

satisfies the Riccati equation

$$\frac{d}{dr}Z(r) + Z^2(r) + k^2\left(1 - \frac{V(r)}{E} - \frac{l(l+1)}{k^2r^2}\right) = 0. \tag{9}$$

At first we solve the problem for the auxillary Potential $\bar{V}$

$$\bar{V}(r,\rho) = V(r)\theta(\rho - r) + V_c(r)\theta(r-\rho), \tag{10}$$

where V_c is the acting the Coloumb potential. Thus the wave function takes the form:

$$\bar{\psi}_l(r,\rho) = \begin{cases} \psi_l(r) & r \leq \rho \\ \alpha_l(\rho)\left[cos\delta_l(\rho)F_l(kr) + sin\delta_l(\rho)G_l(kr)\right] & r \geq \rho \end{cases}. \tag{11}$$

Here, F_l and G_l are known to be the Coulomb functions of the first and second kind. Their properties can be found in [1]. Inserting the wave function for $r \geq \rho$ into (8) one obtains the expression

$$\bar{Z}_l(r,\rho) = \begin{cases} Z_l(r) & r \leq \rho \\ k\frac{cos\delta_l(\rho)F_l'(kr)+sin\delta_l(\rho)G_l'(kr)}{cos\delta_l(\rho)F_l(kr)+sin\delta_l(\rho)G_l(kr)} & r \geq \rho \end{cases}. \tag{12}$$

The logarithmic derivative, $\bar{Z}$ must satisfy the continuity condition. Thus by reinserting $\bar{Z}$ at $r=\rho$ into (9) we obtain

$$\begin{aligned} &\frac{d}{dr}\left[k\frac{cos(\delta_l(r))F_l'(kr) + sin(\delta_l(r))G_l'(kr)}{cos(\delta_l(r))F_l(kr) + sin(\delta_l(r))G_l(kr)}\right] + \\ &\quad + k^2\left[\frac{cos(\delta_l(r))F_l'(kr) + sin(\delta_l(r))G_l'(kr)}{cos(\delta_l(r))F_l(kr) + sin(\delta_l(r))G_l(kr)}\right]^2 + \\ &\quad + k^2\left[1 - \frac{V(r)}{E} - \frac{l(l+1)}{k^2r^2}\right] = 0. \end{aligned} \tag{13}$$

From this expression one gets by evaluating the derivatives explicitly,

$$\begin{aligned} &k^2\frac{cos(\delta)F'' + sin(\delta)G''}{cos(\delta)F + sin(\delta)G} + \\ &\quad + k\frac{d\delta}{dr}\frac{-sin^2(\delta)F'G + cos^2(\delta)FG' - cos^2(\delta)F'G + sin^2(\delta)FG'}{(cos(\delta)F + sin(\delta)G)^2} + \\ &\quad + k^2\left[1 - \frac{V}{E} - \frac{l(l+1)}{k^2r^2}\right] = 0. \end{aligned} \tag{14}$$

Using the Schrödinger equation for the Coulomb potential and inserting (11) into (7) yields

$$\frac{cos(\delta)F'' + sin(\delta)G''}{cos(\delta)F + sin(\delta)G} = -k^2 + \frac{l(l+1)}{r^2} + \frac{V_c}{E}. \tag{15}$$

With (15) and the identity $G'F - F'G = -1$ we obtain from equation (14) the differential equation for the phase shift.

$$\frac{d\delta}{dr} = -k\frac{V(r) - V_c(r)}{E}\left[cos(\delta_l(r))F_l(kr) + sin(\delta_l(r))G_l(kr)\right]^2. \tag{16}$$

For neutron- nucleus scattering the corresponding differential equation is,

$$\frac{d\delta_l}{dr} = -k\frac{V_p}{E}\left[cos(\delta_l(r))\hat{j}_l(kr) - sin(\delta_l(r))\hat{n}_l(kr)\right]^2, \tag{17}$$

where V_p is contributed due to the polarizability as defined in equation (5), $\hat{j}_l(x)$ and $\hat{n}_l(x)$ are the spherical Bessel and Neumann functions in Ricatti form which are known to be $\hat{j}_l(x) = \frac{j_l(x)}{x}$ and $\hat{n}_l(x) = \frac{n_l(x)}{x}$. The solution to our problem can now be found by integrating (17).

4 Numerical Integration using Runge Kutta

We use a Runge Kutta algorithm to solve the non-linear first order differential equation associated with the Variable Phase Method (17). In particular we use a fifth- with embedded fourth- order algorithm which includes variable step size control as discribed by W. H. Press an S. A. Teukolsky [2]. The fifth- order Runge Kutta formula with Cash Karp parameters (Table 1) is given by

$$\begin{aligned} k_1 &= \Delta x f(x_n, y_n), \\ k_2 &= \Delta x f(x_n + a_2\Delta x, y_n + b_{21}k_1), \\ &\dots \\ k_6 &= \Delta x f(x_n + a_6\Delta x, y_n + b_{61}k_1 + \cdots + b_{65}k_5), \quad (18)\\ y_{n+1} &= y_n + c_1k_1 + c_2k_2 + \cdots + c_6k_6 + O(h^6). \quad (19) \end{aligned}$$

The embedded fourth-order formula has the form

$$y^*_{n+1} = y_n + c^*_1k_1 + c^*_2k_2 + \cdots + c^*_6k_6 + O(h^6). \tag{20}$$

Thus one can estimate the error by

$$\Delta = y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6}(c_i - c^*_i)k_i. \tag{21}$$

The estimated error allows us either to accept the step size, or to truncate the entire step if Δ was to large. Because the estimated error scales with $(\Delta x)^5$ the stepsize for the next step is given by

$$\Delta x_{n+1} = \Delta x_n\left|\frac{\epsilon}{\Delta}\right|^{1/5}, \tag{22}$$

a_i	b_{ij}					c_i	c_i^*
						$\frac{37}{378}$	$\frac{2825}{27648}$
$\frac{1}{5}$	$\frac{1}{5}$					0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$

Table 1: Cash Karp parameters for the embedded Runge Kutta method

where ϵ is the requested accurracy for the Runge Kutta algorithm. For $|\epsilon/\Delta|^{0.2} < 0.8$ we truncate the current step and decrease Δx_n to Δx_{n+1} to maintain a a requested accuracy.

5 Numerical Evaluation of the spherical Bessel Function

It is very important for the solution that a very efficient algorithm is used to evaluate the spherical Bessel functions of the first and second kind (also known as spherical Neumann functions). This is particularly important since these functions are called by every Runge Kutta iteration step. The implementation we used follows close to the one described by E. Gillman and H. R. Fiebig [3]. One should note that other than [3] we are not evaluating the spherical Bessel and Neumann functions for multiple l simultanuously. Our program would not benefit from such a procedure as we have to calculate the functions for multiple arguments of the same l.

To evalute the bessel function we use the recurrence relation.

$$g_{l-1}(x) + g_{l+1}(x) = \frac{2l+1}{x} g(x) \tag{23}$$

Which is satisfied for both kinds of the spherical Bessel functions. It is well known that it is possible to calculate spherical Bessel function of the second kind using forward recurrence, however this approach fails for the spherical Bessel functions of the first kind, where backward recurrence has to be used. To avoid underflow or overflow during the calculation of the spherical Bessel function one has to split off the following factors,

$$j_l(x) = \frac{x^l}{(2l+1)!!} u_l(x), \tag{24}$$

$$n_l(x) = -\frac{(2l-1)!!}{x^{l+1}} v_l(x). \tag{25}$$

Hence, the recurrence relations are reduced to those for u_l and v_l

$$u_{l-2}(x) = u_{l-1}(x) - \frac{x^2}{(2l-1)(2l+1)} u_l(x), \tag{26}$$

$$v_{l+1}(x) = v_l(x) - \frac{x^2}{(2l-1)(2l+1)} v_{l-1}(x). \tag{27}$$

To maintain an acceptable accurracy ϵ of the numerically evaluated spherical Bessel function of the first kind a given minimum number L of steps have to be recurred. According to [3] the number L of steps should satisfy

$$L \geq \frac{1}{2}\sqrt{(x^2\epsilon^{1/3} + 9)}. \tag{28}$$

The initial values for the downward recursion are chosen to be 0 and 1. After the downward recursion the values of the functions must be renormalized. This is done using the identity

$$j_l(x)n_{l-1}(x) - j_{l-1}(x)n_l(x) = \frac{1}{x^2}. \tag{29}$$

Which in terms of u_l and v_l becomes

$$u_{l-1}(x)v_l(x) - \frac{x^2}{(2l-1)(2l+1)} u_l(x)v_{l-1} = 1. \tag{30}$$

Thus the normalization factor a according to $\bar{u}_l(x) = au_l(x)$ is given by

$$a = \bar{u}_0(x)v_1(x) - \frac{x^2}{3}\bar{u}_1(x)v_0(x), \tag{31}$$

where $\bar{u}_0(x)$ and $\bar{u}_1(x)$ are the results of the downward recursion. $v_0(x)$ and $v_1(x)$ are as is easily verfied by (25) given by

$$v_0(x) = cos(x), \tag{32}$$

$$v_1(x) = cos(x) + xsin(x), \tag{33}$$

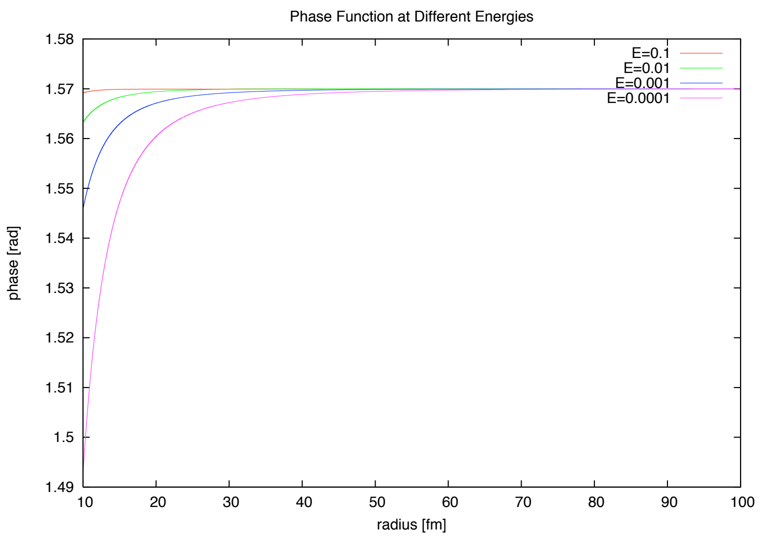
which are the initial values for upward recurrence when evaluating the spherical Neumann functions.

6 Results

So far we explained in great detail how we calculated the phase shifts. In this section we present the results for the potential term (5) caused by the neutron's polarizability. In all calculations we considered a nucleus with $Z = 82$ and the massnumber $A = 208$. Further we set the the neutron's polarizability to $\alpha_n = 10^{-4} fm^3$. The best value found in the literature [5] is $\alpha_n = (1.20 \pm 0.35) 10^{-3} fm^3$.

In Figure 1 we present the radial dependence of the phase shifts $\delta_l(r)$ at $l = 0$ and different energies E. To outline the dependence on the value of the phase shift we show the function for two asymptotic values, i.e. $\delta(\infty) = \frac{\pi}{2}$ and $\delta(\infty) = \pi$.

For an improved visualisation of the dependence of $\delta_l(r)$ we generated contour plots (Figure 2. and 3.). In these plots changes are displayed in the range from $r = 100 fm$ to $r = 10 fm$ at different energies. The size of changes are characterized by colours. Several contour plots show peculiar lines. It turns out that these are artefacts related to limited accuracy of the calculation. The lines vanish by improving the accuracy.



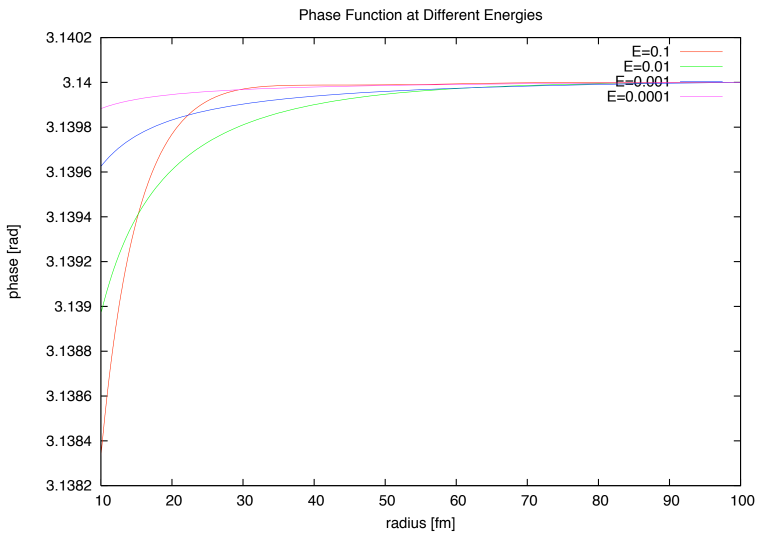


Figure 1: Phase function $\delta(r)$ for different energies. Shown here is the radial dependence of the s-wave phase shift $\delta(r)$ at different asymptotic values $\delta(\infty)$ and energies E.

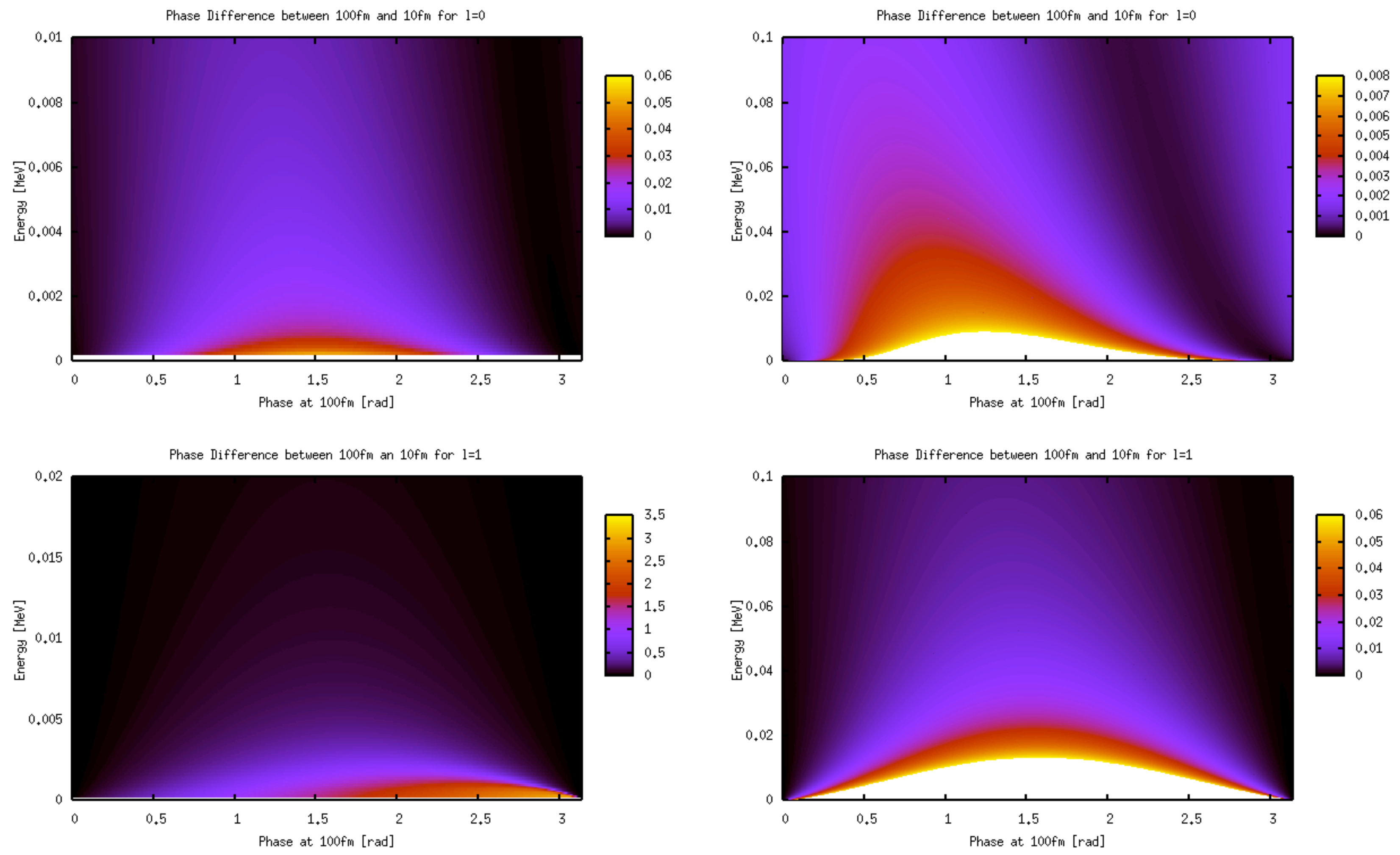


Figure 2: The difference in phase between 100fm and 10fm for $l = 0$ and $l = 1$ scattering. In the two left plots low enrgies are observed as these tend to generate high changes for $\delta(r)$ that do not fit onto the plots on the right side (white areas). We further observe that changes do increase for higher l values.

7 Notes on Accuracy

The contour plots exhibit peculiar patterns of lines for evaluations performed with limited accuracy accuracy ϵ (refer to equation (22)). The apperance of these patterns seems to be quite chaotic. We further found out that the lines disappear at an accuracy of $\epsilon = 10^{-8}$. To demonstrate the effect, we show in Figure 4. contour plots for $l = 0$, one with $\epsilon = 10^{-8}$ and one with $\epsilon = 10^{-6}$.

8 Program Useage

As pointed out in the introduction we wrote the program in Fortran 77 to gain maximum compatability with other programs in the field. The entire algorithm has been embedded into a subroutine, so that it can be called from a master program.

To call the subroutine one may use the following statement in his code:

```
  call runge(begin,theend,init,initdelta,acc,l,Z,polarizeability,
 +    E,mtarget,min)
```

The arguments and their formats are discribed in Table 2.

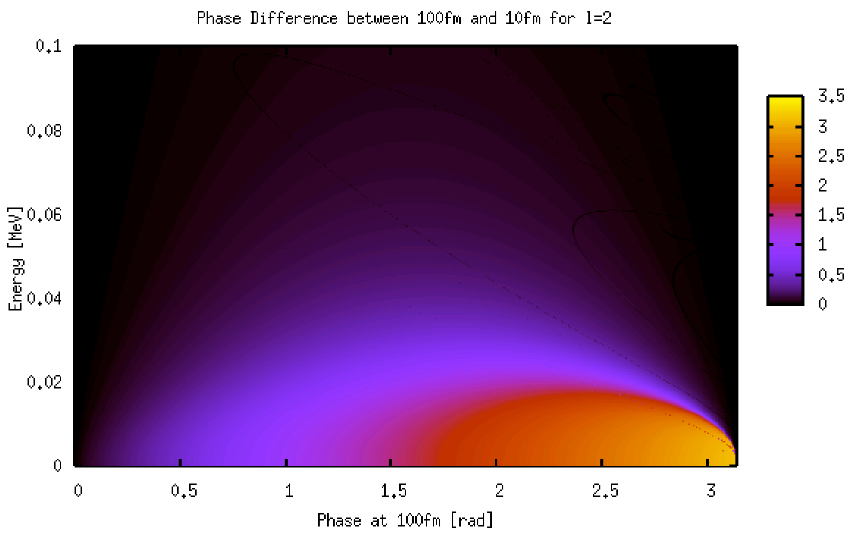
The program writes the values for the phase function to *stdout* which may be redirected to a file from where one can plot the scattering phase. The first value on each line from the output gives the r-value, the second the phase value and the third represents the factor which is multiplied to the current step size to correct the step size in the next step. The third value, more or less meaningless for the result, is provided so that one can monitor the adaptive step size control of the Runge Kutta algorithm.

For the generation of the contour plots the program has been slightly modified. A patch file should be provided with the program so that one can generate these plots.

9 Runtime Estimation

In the following we consider the runtime of the program, in order to provide data for a reliable estimate of the required runtime. We further point out that the estimations given, may not be accurate due to the size of parameters that effect the runtime of todays computer systems. The only component which we use for our estimation is the CPU-time, which probably is the most important one for such an evaluation.

The system we used for our calculations is a PowerMac G5 featuring two IBM PPC 970 CPUs clocked at 2GHZ. Our program utilises only one CPU



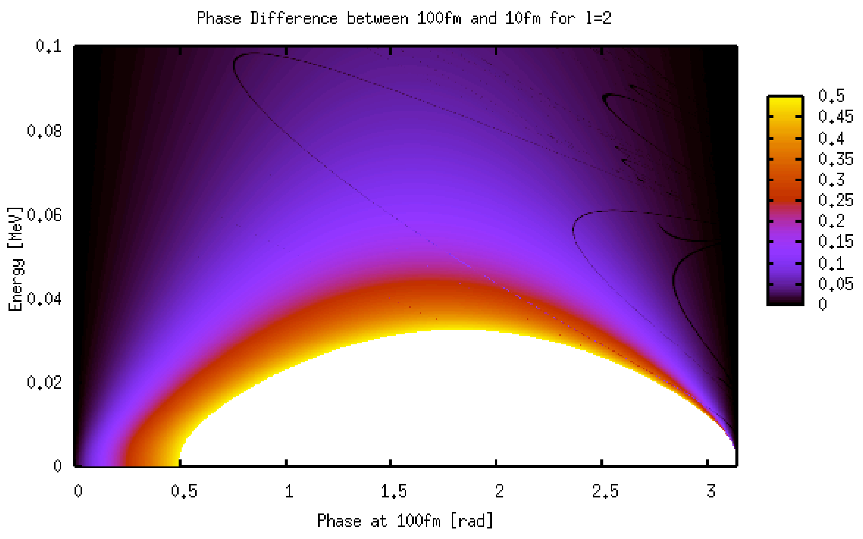
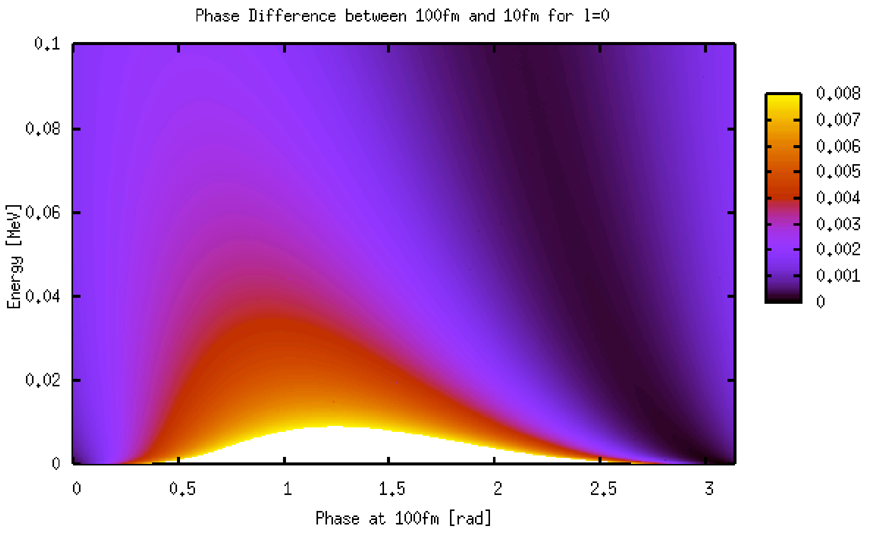


Figure 3: The difference in phase between 100fm and 10fm for $l = 2$ scattering.



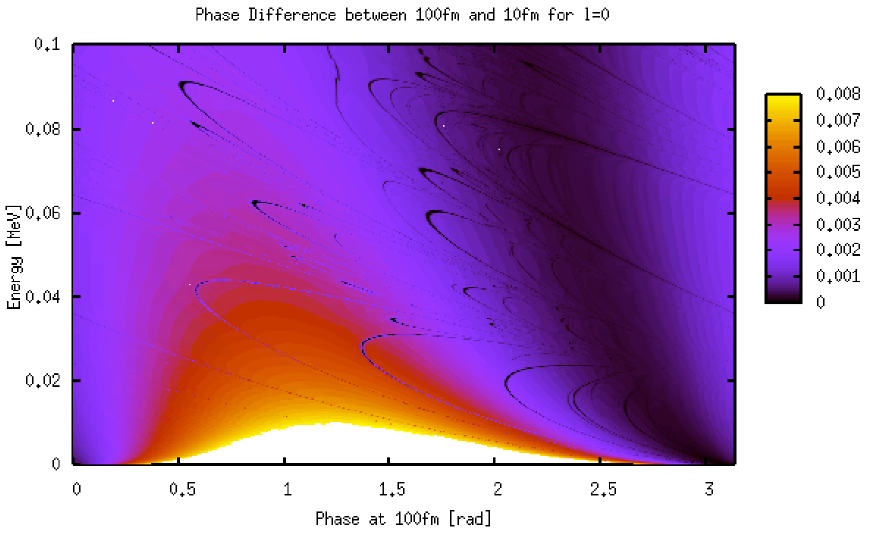


Figure 4: The upper plot was created with an accuracy of $\epsilon = 10^{-8}$. The lower plot with an accuracy of $\epsilon = 10^{-6}$.

Name	Description	Data Type	Units
begin	Radius to start from	double precision	fm
theend	Radius where to stop	double precision	fm
init	Scattering phase at begin	double precision	radian
initdelta	Initial step size†	double precision	fm
acc	Required Accuracy††	double precision	
l	Angular Momentum	integer	
Z	Charge of the nucleus	integer	e^+
polarizeability	Polarizeability of the Neutron	double precision	fm^3
E	Energy	double precision	MeV
mtarget	Mass of the nucleus	double precision	AMU
min	Mass of the neutron	doubel precision	AMU

Table 2: Arguments of the subroutine that calls our program. † has to be negative; †† refere to equation (21) and (22) to understand how this value is used

during runtime. The operating system installed is Mac OS X 10.4.7. We compiled our program with g77, part of gcc 3.4, with the following optimisation flags,

```
-O3 -falign-loops=16 -falign-jumps=16 -falign-functions=16
-ffast-math -mtune=970 -mcpu=970 -mpowerpc-gfxopt
-pipe -fomit-frame-pointer
```

To measure the runtime we consider the evaluations for the contour plots. A resolution of 1000x1000 pixels was chosen for the contour plots. Thus to evaluate one contour plot the computer has to evalutate the phase function 10^6 times. For $l = 0$ scattering with an accuracy of $\epsilon = 10^{-8}$ this took us about 45 minutes. For $l = 1$ scattering at the same accuracy it took us about 15 hours and 15 minutes. We were not able to evaluate $l = 2$ scattering at this accuracy.

As one can see the computational effort for solving the phase function increases dramatically for higher l. This is caused by two things. First according to (28) the computational effort to evaluate the spherical Bessel function increases. Secondly the difference in phase is higher and thus the absolute value of the first derivative (17) is higher, which causes the step size control in the Runge Kutta algorithm to reduce the stepsize dramatically resulting in more steps to be evaluated.

The average time for one evaluation of the phase function is given by $t_0 = 0.0027$sec for $l = 0$ and by $t_1 = 0.055$sec for $l = 1$ at an accuracy of $\epsilon = 10^{-8}$. Thus one may estimate the time to solve a problem consisting of

n evaluations of the phase functions on a machine close to ours by,

$$t = t_i n. \tag{34}$$

We remark that different energies and masses may affect the effective runtime. Both influence the derivative of the phase shift $\frac{d\delta}{dr}$ and thus the value of the chosen step size.

10 Program Listing

The listing of the main program is presented in this section. The program has to be compiled with or linked together with the object files of the codes evaluating the spherical Bessel and Neumann functions. The codes for the spherical Bessel and Neumann functions are also listed.

```
c     Subroutine that solves the differential equation for the
c     phase function using an adaptive stepsize runge kutta algorithm.

      subroutine runge(beginvalue,endvalue,initialvalue,initialdelta,
     +     accuracy,l,Z,polarizeability,E,mtarget,min)

      double precision r,endvalue,beginvalue,initialvalue,initialdelta,
     +     delta,k(6),errorest,phase,accuracy,check,newphase,
     +     polarizeability,E,mtarget,min,mreduced,kmom

      integer l,Z

      r=beginvalue
      phase=initialvalue
      delta=initialdelta

c     The reduced mass is given by
      mreduced=(mtarget*min)/(mtarget+min)

c     k is given by k=sqrt(2*mreduced*c**2*E/(hbar**2*c**2))
      kmom=dsqrt((2*mreduced*931.49*E)/(32041.))

c     Begin of Runge Kutta Iteration Loop

c     In case you want to iterate outwards ( not inwards as we do )
c     the following line to
c10   if (r.lt.endvalue) then
```

```
c     initialdelta has to be positive for this

 10   if (r.gt.endvalue) then

         k(1) = delta*newphase(r,phase,l,Z,polarizeability,kmom,E)
         k(2) = delta*newphase(r+0.2*delta,phase+0.2*k(1),l,
     +        Z,polarizeability,kmom,E)
         k(3) = delta*newphase(r+0.3*delta,phase+0.075*k(1)+
     +        0.225*k(2),l,Z,polarizeability,kmom,E)
         k(4) = delta*newphase(r+0.6*delta,phase+0.3*k(1)-0.9*k(2)+
     +        .2*k(3),l,Z,polarizeability,kmom,E)
         k(5) = delta*newphase(r+delta,phase-11./54.*k(1)+1.5*k(2)+
     +        -70./27.*k(3)+15./27.*k(4),l,Z,polarizeability,kmom,E)
         k(6) = delta*newphase(r+0.875*delta,phase+1631./55296.*k(1)+
     +        175./512.*k(2)+575./13824.*k(3)+44275./110529.*k(4)+
     +        235./4096.*k(5),l,Z,polarizeability,kmom,E)


         errorest = (37./378.-2825./27648.)*k(1)+
     +        (250./621.-18575./48384.)*k(2)+
     +        (125./594.-13525./55296.)*k(3)-
     +        227./14336.*k(4)+
     +        (512./1771.-1./4.)*k(5)


c     Stepsize adaption
         check = dabs(accuracy/errorest)
         delta = delta*(check**0.2)

c     Check whether the steps accuracy condition is met
         if (check.gt.0.8) then
            phase = phase+37./378.*k(1)+250./621.*k(2)+125./594.*k(3)+
     +           512./1771.*k(6)

            write(*,*) r, phase, check**0.2
            r=r+delta
         endif

         goto 10

      endif

      end
```

```
c     The function to be integrated
      double precision function newphase(r,phase,l,Z,polarizeability,
     +     kmom,E)
      double precision r,phase,sphneumann,sphbessel, polarizeability,
     +     kmom,E,kmomr
      integer l,Z
      kmomr = r*kmom
c     In our units (hbar*alphaf*c)/2 = 1.4379
      newphase = -(Z**2.)*polarizeability*1.4379*1./(r**4.)*(-kmom/E)*
     +     (dcos(phase)*sphbessel(kmomr,l)*kmomr-
     +     dsin(phase)*sphneumann(kmomr,l)*kmomr)**2.

      return
      end
```

The code for the spherical Bessel function:

```
c
c     Fortran Function to evaluate the spherical bessel function
c
c     with spilt off of ill behaved factors as described by E.Gillman H.R.
c     Fiebig

      double precision function sphbessel(r,l)

      double precision j(3),testa,testb,r,unnormalizedvalue,
     +     normalizationfactor


      integer l,k,precision,lu,doublefact

c     where (10E-7)**(-1/3) is approx 215.4435 thus
c     the precision is given by epsilon 10**-6 at least
      precision=dint(0.5*((r**2.*215.4435+9.)**0.5)+1.)

      k=l+1

      j(3)=0
      j(2)=1

      if(k.eq.1) then
```

```
          sphbessel=dsin(r)/r
          return

        else

           do i=1,k+precision

              lu=k+precision-i+2
              j(1)=j(2)-(r**2.)/(4.*lu**2.-1.)*j(3)

              if (lu.ne.2) then
                 j(3)=j(2)
                 j(2)=j(1)
              endif

             if(k+precision-i+1.eq.k) then
                 unnormalizedvalue=j(1)
              endif

           enddo

           doublefact=1
           do i=1,2*l+1,2
              doublefact=doublefact*i
           enddo

           testa=dcos(r)
           testb=dcos(r)+dsin(r)*r
           normalizationfactor=j(1)*testb-(r**2.)/3.*j(2)*testa
           sphbessel=(r**l)/doublefact*
     +          unnormalizedvalue/normalizationfactor
           return

        endif

        end
```

The code the spherical Neumann function:

```
C     Spherical Neumann functions
C     Calculated using forward reccurrence
```

```
C     Splitting off ill behaved factors according to E. Gillman, H. Fiebig

      double precision function sphneumann(r,l)

      double precision n(3),r
      integer l,k

      k=l+1

      n(1) = dcos(r)
      n(2) = dsin(r)*r+dcos(r)


      if (k.gt.2) then

c     Beginn of recursion loop

         do i=3,k
            n(3)=n(2)-(r**2)/(4.*(i-2)**2-1)*n(1)
            n(2)=n(3)
            n(1)=n(2)
         enddo
      else
         if(k.eq.2) then
            n(3)=n(2)
         else
            n(3)=n(1)
         endif
      endif

      doublefact=1
      do i=1,2*l-1,2
         doublefact=doublefact*i
      enddo

      sphneumann=-n(3)*doublefact/(r**(l+1))

      return
      end
```

Patch to generate contour plots (only prints out the last value from Runge Kutta iteration). The program has to be embedded into a loop running through different energy and asymptotic phase values $\delta(\infty)$ in order to generate plots like those shown in this document.

```
*** variablephase.f     Thu Jul 20 21:06:51 2006
--- variablephase-loop.f        Thu Jul 20 21:06:49 2006
***************
*** 59,72 ****
              phase = phase+37./378.*k(1)+250./621.*k(2)+125./594.*k(3)+
       +           512./1771.*k(6)

!             write(*,*) r, phase, check**0.2
              r=r+delta
           endif

           goto 10

        endif
!
        end

  c     The function to be integrated
--- 59,72 ----
              phase = phase+37./378.*k(1)+250./621.*k(2)+125./594.*k(3)+
       +           512./1771.*k(6)

!
              r=r+delta
           endif

           goto 10

        endif
!       write (*,*) initialvalue , E, phase, dabs(initialvalue-phase)
        end

  c     The function to be integrated
```

References

[1] *M. Abramowitz, A. Stegun: Handbook of Mathematical Functions* (Dover Publications, New York, 1974).

[2] *W. H. Press and S. A. Teukolsky: Adaptive Stepsize Runge-Kutta Integration*, Computers in Physics, 2, 188 (1992).

[3] *E. Gillman, H.R. Fiebig: Accurate recursive generation of spherical Bessel functions for a large range of indices*, Computers in Physics, 1, 62 (1988).

[4] *F. Calogero: Variable Phase Approach to Potential Scattering* (Academic Press, New York and London, 1967).

[5] *J. Schmiedmayer, P. Riehs, J. A. Harvey, N. W. Hill: Measurement of the Electric Polarizablity of the Neutron*, Phys. Rev. Lett. 66, 8, (1991).