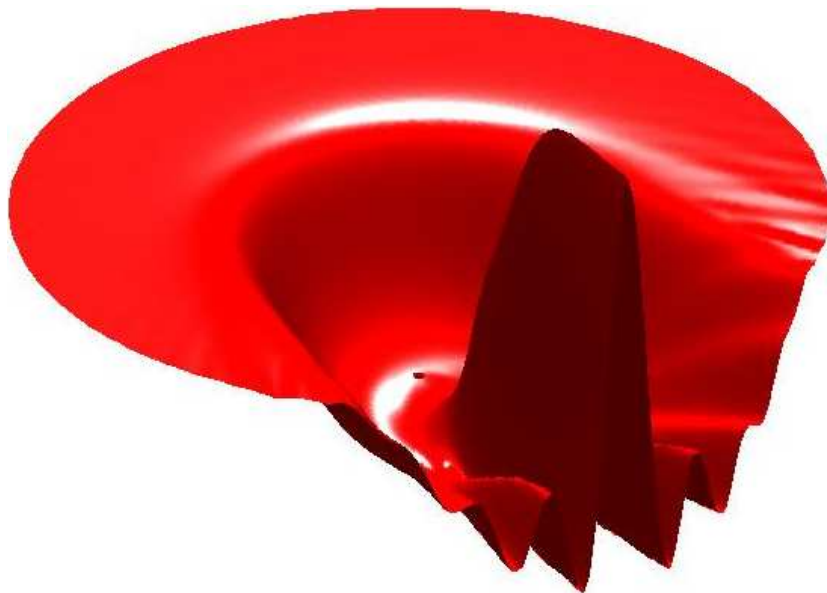


The course Kvantmekanik, fk is offered in the fourth year of the physics engineering program (Teknisk fysik, inriktning Tillämpad fysik) and the Natural Science program (Naturvetarprogrammet, inriktning fysik) at Uppsala University. It is based on the text *Modern Quantum Mechanics* by J. J. Sakurai (Addison-Wesley Publishing Company, Revised Edition 1994). Numbered equations below refer to this text.

Instructions for a computer based study of

Quantum Scattering in 3D



0. Introduction

Quantum scattering in 3 dimensions will be studied through numerical solution (using MATLAB) of the Schrödinger equation for a nuclear physics model problem, the scattering of an alpha particle off the nucleus ^{12}C . The interaction potential is assumed to have a Woods-Saxon form,

$$V(r) + iW(r) = V_0 \frac{1}{1 + e^{(r-R_V)/a_V}} + iW_0 \frac{1}{1 + e^{(r-R_W)/a_W}}$$

In this potential, the imaginary part simulates the loss of particles from the elastic channel due to various nuclear reactions. Properly, the Coulomb interaction should also be included, but this is not done here.

It is a primary goal of nuclear phenomenology to determine the parameters of the potential from measurements of the differential cross section $d\sigma/d\Omega$. For this purpose, $d\sigma/d\Omega$ is calculated for a range of parameters, and the set that gives the best fit to the experimental numbers is selected. Here, it will be verified that such a selected set of parameters¹ indeed reproduces the measured values well.

The study has three parts.

1. In the first, quantum scattering concepts are introduced and studied, and the MATLAB code is provided.
2. In the second part, the Schrödinger equation is solved in detail, starting with an interaction potential, and ending up with a differential cross section which is compared to experimental data. In this step, some MATLAB code is provided, and some is suggested, but the detail is such that at the end you should be able to say: I know how to solve the Schrödinger equation in 3 dimensions on the computer, for a simple but realistic case.
3. In the third part, further insight into quantum scattering is offered. For this part, a somewhat more elaborate set of code has been combined with a graphic user interface into a MATLAB package named *qmehc*, thus allowing a more streamlined investigation. The highlight is a movie showing the scattering of a 3D quantum wavepacket.

¹curtesy A. Ingemarsson, IKP

1. Quantum scattering concepts

1.1 The shape of the potential

Open the package *qmech* by entering *qmech* at the MATLAB prompt. Push the button '*Scattering in 3D*' and then the button '*Potential*'. Chose alpha and carbon-12 as your particles.

IMPORTANT:

switch off the Coulomb potential by hand
by putting one of the charges equal to 0.

For potential, choose '*Woods-Saxon*'. The parameters of the potential can be changed by typing new values in the editable fields, or by using the sliders. Play with the parameters, and answer the following three questions (hint: use words like range, depth, and diffuseness):

Q1.1. Which feature of the potential is affected if the parameters V_0 and W_0 are varied?

Q1.2. Which feature of the potential is affected if the parameters R and a are varied?

Q1.3. How can the potential be made to look like a spherical well with a flat bottom and a very sharp boundary?

Before leaving this window, set the parameters to correspond to the alpha-¹²C system at 129 MeV center of mass energy (enter *alphaC12par* at the MATLAB prompt),

$$\begin{array}{ll} V_0 = -116.51 & \text{MeV} & W_0 = -16.94 & \text{MeV} \\ R_V = 2.559 & \text{fm} & R_W = 4.165 & \text{fm} \\ a_V = 0.8313 & \text{fm} & a_W = 0.5277 & \text{fm} \end{array}$$

Question: Is this a very deep potential?

Answer: Not particularly, considering the total energy.

Question: Why is this type of potential referred to as an optical potential?

Answer: It has the same shape as the index of refraction of a (not entirely transparent) ball of glass.

Question: Is the quantum scattering from this type of potential similar to the scattering of light from such a glass ball?

Answer: Yes, to quite some extent!

1.2. Is the scatterer big or small?

In any wave scattering problem, it is useful to know if the wavelength is larger or smaller than the size of the scatterer.

Find the wavelength corresponding to alpha-¹²C scattering at 172.5 MeV alpha lab energy, and compare this value to the extension of the potential.

For this purpose, prepare the MATLAB m-file *M1_wavenumber.m* according to the code below.

MATLAB	code for <i>M1_wavenumber.m</i>	MATLAB comments
clear		
hbarc	= 197.329	% $\hbar c$ in Mev fm
m1	= 3727.3	% alpha mass in MeV
m2	= 11175	% ¹² C mass in MeV
mred	= m1*m2/(m1+m2)	% the reduced mass
Elab	= 172.5+[-160:160];	% lab energy in MeV
Ecm	= m2/(m1+m2)*Elab;	% cm energy in MeV
k_all	= sqrt(2*mred*Ecm)/hbarc;	% wave number in fm ⁻¹
		% Eqn A.5.11
lambda_all	= 2*pi./k_all;	% wavelength in fm
k	= k_all(161)	
lambda	= lambda_all(161)	
save	sparring hbarc mred k	% save for later use
%The rest is plotting		
figure		
plot(Elab, lambda_all)		
title('wavelength as fcn of Elab')		
clear		

Run the code by entering *M1_wavenumber* at the MATLAB prompt.

Q1.4 What do you find for the wave number and the wavelength?

Q1.5 Compare the wavelength to the extension of the potential. Why is this an interesting thing to do? What kind of phenomena do you expect if the wavelength is about the size of the diameter of the potential well? (If you have time, you may investigate this point further in part 3.)

1.3. The Schrödinger equation

For a two-body system, the relative motion Schrödinger equation has the form

$$\left(-\frac{\hbar^2}{2\mu}\nabla^2 + V(r) + iW(r) - E\right)\psi(\vec{r}) = 0 \quad (1)$$

where μ is the reduced mass. In a scattering problem one is interested in solutions that at large r behave like (Eqn 7.1.33)

$$\psi(\vec{r}) \rightarrow N \left(e^{i\vec{k}\cdot\vec{r}} + f(\vec{k}', \vec{k}) \frac{e^{ikr}}{r} \right)$$

The plane wave part $e^{i\vec{k}\cdot\vec{r}}$ corresponds to a probability current (Eqn 2.4.16)

$$\vec{j}(\vec{r}) = \frac{\hbar}{\mu} \text{Im} \left(\psi^*(\vec{r}) \nabla \psi(\vec{r}) \right) = \frac{\hbar \vec{k}}{\mu} \rho(\vec{r}),$$

that flows in the \hat{k} direction, and represents an incoming particle with momentum $\vec{p} = \hbar \vec{k}$ and energy $E = p^2/2\mu$ (properly, one should introduce wave packets, etc).

The second part is associated with a probability current (at $r \rightarrow \infty$)

$$\vec{j}(\vec{r}) = \frac{\hbar k}{\mu} \frac{\hat{r}}{r^2} |f|^2 \rho(\vec{r})$$

that flows out from the scattering region, and represents the scattered particles. It gives rise to a detection rate in a distant detector that depends on the *scattering amplitude* f . If the rate is normalized to the incoming flux, one is left with the *differential cross section*

$$\frac{d\sigma}{d\Omega} = |f|^2$$

as a measure of the scattering ability of the potential (Eqn 7.1.36).

Typically, the results of scattering experiments are reported in terms of this quantity.

A major objective of the present study is to see how f and the differential cross section can be obtained from the numerical solution of the Schrödinger equation.

Often, and as is the case here, $f(\vec{k}', \vec{k})$ and $\frac{d\sigma}{d\Omega}$ only depend on the scalar $\vec{k}' \cdot \vec{k}$, but not on the direction vector given by $\vec{k}' \times \vec{k}$.

Q1.6 If $\vec{k} = \hat{z}$ and $\vec{k}' = k \sin \theta \cos \phi \hat{x} + k \sin \theta \sin \phi \hat{y} + k \cos \theta \hat{z}$, why doesn't $f(\vec{k}', \vec{k})$ depend on the azimuthal angle ϕ ?

For the alpha-¹²C system at 172.5 MeV alpha lab energy, experimental differential cross sections, converted to center-of-mass quantities, are available for scattering angles between 6 and 60 degrees. Look at the experimental results by typing at the MATLAB prompt

MATLAB code	MATLAB comments
<code>>> plot_data</code>	<code>% Experimental data</code>

Note the logarithmic scale, i.e. be sure to appreciate how fast the cross section drops when the detector is moved out from the forward direction!

Q1.7 Suppose it takes 1 min to get N counts in a detector at 6 degrees. How long time will it take to get the same number of counts at 60 degrees?

2. From potential to cross section

2.1. How to solve a PDE in 3 variables?

Since the form of the Schrödinger equation is rotationally invariant, it is natural to introduce spherical coordinates (Sakurai, Appendix 5). The solutions, however, will in general not be spherically symmetric (why?), and it is useful to expand them in a suitable complete set of angular functions.

Question: Which angular functions are the best?

Answer: The eigenfunctions of (the angular part of) ∇^2 , i.e. the spherical harmonics $Y_l^m(\theta, \phi)$, since in this case the resulting ordinary differential equations for the expansion coefficients become the simplest (i.e. not coupled)!

Also the boundary condition must be expanded in the same set of angular functions. But the boundary condition is cylindrically symmetric. Therefore, the simplest expansion is obtained if the polar axis of the spherical coordinates is chosen along the symmetry axis, i.e. along the direction of the incoming current, \hat{k} . In such a case, the incoming plane wave depends only on the polar angle θ and not on the azimuthal angle ϕ , and in the expansion in spherical harmonics only terms with $m = 0$ contribute,

$$Y_l^0(\theta, \phi) = \sqrt{\frac{2l+1}{4\pi}} P_l(\cos \theta)$$

(Eqn 7.6.4), i.e. the expansion of the incoming plane wave reduces to an expansion in Legendre polynomials (Eqn 7.5.18),

$$e^{ikr \cos \theta} = \sum_{l=0}^{\infty} i^l (2l+1) j_l(kr) P_l(\cos \theta). \quad (2)$$

where $j_l(kr)$ denotes a spherical Bessel function. As a consequence, the expansion of the full solution to the Schrödinger equation can also be limited to an expansion in Legendre polynomials in $\cos \theta$,

$$\psi(r, \theta) = \sum_{l=0}^{\infty} i^l (2l+1) \frac{u_l(r)}{r} P_l(\cos \theta) \quad (3)$$

In the next step, the radial wave functions $u_l(r)$ are to be determined. First, however, a short digression on Bessel functions and Legendre polynomials.

2.2. Spherical Bessel and Legendre polynomials

In the package *qmech* there are MATLAB m-functions *s_code34* and *s_code33* defined that generate plots of the spherical Bessel functions $j_l(x)$ and $y_l(x)$ ($= n_l(x)$).

MATLAB code	MATLAB comments
>> <code>s_code34</code>	% $j_l(x)$
>> <code>s_code33</code>	% $y_l(x)$

Run these m-functions, look at the plots, consult the textbook (Sakurai, Appendix 5), and answer the question:

Q2.1 How do $j_l(x)$ and $n_l(x)$ behave when $x \rightarrow 0$ and when $x \rightarrow \infty$ (i.e. how fast does $j_l(x) \rightarrow 0$, etc.)?

The Legendre polynomials are displayed by the m-function *s_code31*. Run this code,

MATLAB code	MATLAB comments
>> <code>s_code31</code>	% $P_l(x)$

and answer the question:

Q2.2 Which are the values of $P_l(1)$ and $P_l(-1)$? How many zeros has $P_7(x)$?

You should also investigate the expansion of a plane wave in Legendre polynomials, Eqn (2), and find how fast is the convergence. To proceed with this point, consider the real part of the plane wave expansion, for some fixed r or θ . For simplicity, chose $\theta = 0$ since $P_l(1) = 1$ for all l . The real part of the plane wave expansion then takes the form (with $l = 2n$)

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n (4n + 1) j_{2n}(x)$$

This expansion is illustrated through the code *legendre_1*. Run this code,

MATLAB code	MATLAB comments
>> <code>legendre_1</code>	% code in appendix

and answer the question

Q2.3. How many terms are necessary in order to achieve an accuracy of about 0.1% in the range $0 < x < 25$?

Note that the zoom functionality for Matlab plots can be very useful here. Choosing instead a fixed value for kr , say $kr = 15$, the real part of the plane wave expansion is

$$\cos(15 \cos \theta) = \sum_{n=0}^{\infty} (-1)^n (4n + 1) j_{2n}(15) P_{2n}(\cos \theta)$$

as is illustrated through the code `legendre_2`. Run this code,

MATLAB code	MATLAB comments
<code>>> legendre_2</code>	<code>% code in appendix</code>

and answer the question

Q2.4 How many terms are required in order to achieve an accuracy of about 0.1% for all angles $0 < \theta < \pi$?

2.3. The radial equation.

After expansion of $\psi(r, \theta)$ in Legendre polynomials according to Eqn (3), the Schrödinger PDE, Eqn (1), in two variables has been 'reduced' to an infinite set of (uncoupled) ODEs for the radial wave functions (cf. Eqn A.5.9)

$$\left(-\frac{d^2}{dr^2} + \frac{l(l+1)}{r^2} + \frac{2\mu}{\hbar^2} [V(r) + iW(r)] - k^2 \right) u_l(r) = 0, \quad (4)$$

supplemented with the boundary condition at the origin

$$u_l(0) = 0.$$

We shall now solve the radial Schrödinger equation (4) for some partial waves l and look at the solutions.

In order to make use of the ODE solvers of MATLAB, first convert the radial Schrödinger equation Eqn (4) to first order form by introducing

$$u1 = u \quad u2 = \frac{du}{dr},$$

which allows the differential equations to be written in matrix form like

$$\frac{d}{dr} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ p & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad (5)$$

where

$$p = \frac{l(l+1)}{r^2} + \frac{2\mu}{\hbar^2} (V(r) + iW(r)) - k^2. \quad (6)$$

For $l > 0$, the boundary conditions at the origin require some attention since both $u_l(0) = 0$ and $u'_l(0) = 0$. However, and as is easily verified, $u_l(r) \sim r^{l+1}$ while $u'_l(r) \sim r^l$ for small r , i.e. $u_l(r) \rightarrow 0$ faster than $u'_l(r)$ as $r \rightarrow 0$. A numerical solution can therefore be started at some point $r = r_{\min}$ close to 0, with the conditions $u(r_{\min}) = 0$ and (for an unnormalized solution) $u'_l(r_{\min}) = 1$ or $u'_l(r_{\min}) = r_{\min}^l$.

A template for the MATLAB code for solving the radial Schrödinger equation is provided as *M2_Schrodinger_template.m* and contains code for overhead like overall layout, input/output details, and plotting. The standard solver *ode45* is used, and the details of the differential equation (5) are specified in the MATLAB function *myuprim*.

Create your own m-function *M2_Schrodinger.m* in your work directory, include the template code and supply the missing steps. Make sure to replace *NN* in the legend with your own names! The code for p follows directly from Eqn (6). Note that *ode45* calls *myuprim* at one r -value at a time, i.e. the variables rr and p in *myuprim* are not arrays but simply numbers. The *uprim* on the other hand, like uu , is a column vector, in our case with two components.

MATLAB	code for <i>M2_Schrodinger_template.m</i>	MATLAB comments
function	M2_Schordinger	
load figure	sparning;	% loads hbarc mred k
rmax lmax	= 20; = 60;	% solve out to 20 fm % 61 partial waves
for l = 0:lmax		
rmin	= (l+1)^2/(l+10)/50;	
ustart	= [rmin^(l+1); (l+1)*rmin^l];	% boundary value
options	= odeset('RelTol',1.e-4);	
[r,u]	= ode45(@myuprim, [rmin rmax],... ustart,options,k,l,mred,hbarc);	% solution to Eqn (5)
uend(l+1,:) = u(end,:);		% remember u at rmax
%plotting:		
u(:,1)	= u(:,1)./max(abs(u(:,1)));	% normalize
plot(0,0,r, real(u(:,1)./r), ... r, imag(u(:,1)./r));		% plot radial wave fcn
title(sprintf('Re and Im of R_l(r)= ... u_l(r)/r for l=%3.0f',l))		
xlabel('r'); ylim([-0.5 0.5]);		
legend('NN has just solved the Schröd ... equation!', 'real(R(r))', 'imag(R(r))')		
pause(0.2)		
end		% end of l loop
save sparning lmax rmax uend -append		% save lmax rmax uend
function uprim	= myuprim(rr,uu,k,l,mred,hbarc)	
V0 = -116.51; RV=2.559; aV=0.8313;		% Woods-Saxon
W0 = -16.94; RW=4.165; aW=0.5277;		% parameters, page 3
p = ADD MISSING CODE!		% p from Eqn (6)
uprim	= [uu(2); p*uu(1)];	% two components

The solution to the radial Schrödinger equation can now be generated, for as many l as are needed. Start your calculation by entering `M2_Schrodinger` at the MATLAB prompt and run the code till the end, i.e. for all partial waves.

From the resulting plots it can be seen that it is sufficient to solve the equation for only a small number of l -values.

Q2.5 Can you judge from the plots how many partial waves will turn out significant when generating the cross section? Remember your estimate and compare with the actual result obtained later.

2.4. S-matrix and phase shift

After solving the Schrödinger equation the solution can be matched to the known asymptotical expression of $u_l(r)$ for large r

$$u_l(r) \rightarrow r \left(j_l(kr) + \frac{1}{2i}(S_l - 1)h_l^+(kr) \right) = \frac{r}{2i} \left(S_l h_l^+(kr) - h_l^-(kr) \right) \quad (7)$$

(recall that $h_l^\pm(x) = -n_l(x) \pm ij_l(x)$; note also that the $n_l(x)$ of Sakurai is usually denoted $y_l(x)$ (see Abramowitz and Stegun, *Handbook of Mathematical functions*).

The factor S_l is known as the scattering matrix, or *S-matrix*. It encodes the effect of the scatterer on the outgoing wave in the following sense:

- (i) without the scatterer, $S_l=1$;
- (ii) with a real potential, the outgoing current must equal the incoming current, i.e. $|S_l| = 1$. S_l can then be parametrized in the form

$$S_l = e^{2i\delta_l}. \quad (8)$$

using a *real* phase factor δ_l known as the *phase shift*;

- (iii) for an absorbing potential, the outgoing current may be less than the incoming current. In this case $|S_l| < 1$, and a useful parametrization of the S-matrix is

$$S_l = \eta e^{2i\delta_l} \quad (9)$$

where $0 < \eta < 1$.

Once the wave function has been solved for, the S-matrix can be extracted. Consider the solution $u_l(r)$ at some point $r = R$ well outside the potential, and introduce (Eqn 7.6.34)

$$\beta = r \frac{d}{dr} \left(\frac{u_l}{r} \right) / \left(\frac{u_l}{r} \right) \Big|_R = R \frac{u_2(R)}{u_1(R)} - 1. \quad (10)$$

where $u1$ and $u2$ in the last equality refer to the two components of $u_l(r)$ introduced in connection with Eqn (5). This expression allows β to be computed from the numerical solution of the Schrödinger equation. Using now the analytical expression from Eqn (7) one finds for S-waves, i.e. for $l = 0$, that

$$\begin{aligned} u1(R) &= N(S_0 e^{ikR} - e^{-ikR}) \\ u2(R) &= ikN(S_0 e^{ikR} + e^{-ikR}) \end{aligned}$$

where N is a normalization constant. From Eqn (10) we obtain

$$S_0 = \frac{\beta + 1 + ikR}{\beta + 1 - ikR} e^{-2ikR}$$

For any general l , introduce the notation $h_l'^{\pm}(x) = \frac{d}{dx} h_l^{\pm}(x)$. Then

$$\begin{aligned} u1(R) &= NR \left(S_l h_l^+(kR) - h_l^-(kR) \right) \\ u2(R) &= u1(R)/R + NkR \left(S_l h_l'^+(kR) - h_l'^-(kR) \right) \end{aligned}$$

and by combining these expressions and using Eqn (10)

$$\beta = kR \frac{S_l h_l'^+(kR) - h_l'^-(kR)}{S_l h_l^+(kR) - h_l^-(kR)}.$$

Finally, solving for S_l , we get

$$S_l = \frac{\beta h_l^-(kR) - kR h_l'^-(kR)}{\beta h_l^+(kR) - kR h_l'^+(kR)} \quad (11)$$

(compare Eqn 7.6.35). Recall that $h_l^{\pm}(x) = -n_l(x) \pm ij_l(x)$. By using the numerical values for β obtained from $u1$ and $u2$, S_l can now be determined for each l .

A template for the MATLAB code for finding S_l and δ_l is provided as *M3_Smatrix_template.m*. Create your own m-function *M3_Smatrix.m* in your work directory, include the template code, and supply the missing steps specific for the evaluation of S_l and δ_l according to the equations derived above.

Help concerning the missing code is available in the Appendix.

MATLAB	code for <i>M3_Smatrix_template.m</i>	comments
function	M3_Smatrix	
load	sparning	
L	= 0:lmax;	% L is a vector
x	= k*rmax;	
x1	= x+0.0001;	
hplus	= -sphbessely(L,x)+i*sphbesselj(L,x);	% $h_l^{(+)}(kR)$
hplus1	= -sphbessely(L,x1)+i*sphbesselj(L,x1);	
hplusprim	= (hplus1-hplus)/(x1-x);	% derivative
beta	= ADD MISSING CODE	% from Eqn (10)
S1	= ADD MISSING CODE	% from Eqn (11)
phaseshift	= ADD MISSING CODE	% from Eqn (9)
save	sparning S1 -append;	% save S1
% The rest is plotting		
figure		
plot(L,abs(S1));	title('abs(S_1)'); xlabel('l');	
text(1,0.9,'Push any key to continue');	pause	
figure;		
plot(S1);	axis equal;	
title('S_1 in complex plane.')		
text(-1,-1,'Are all points on the unit circle?');		
set(gca,'NextPlot','add', ...	'Xlim',[-1.2 1.2],'Ylim',[-1.2 1.2]);	
for index = 1:(lmax+1)		
plot(S1(index),'o','Color','red');		
t1 = text(-1,1,sprintf('l=%3.0f',L(index)));		
pause(0.5); delete(t1);		
end;		
text(-1,0.9,'Push any key to continue');	pause	
figure;		
phaseshift =fliplr(unwrap(fliplr(2*phaseshift)))/2;		
plot(L,phaseshift,'o');		
title('phaseshift (rad) as a function of l');		
xlabel('l');	ylabel('\delta_l');	

Start your evaluation of S_l and δ_l by entering *M3_Smatrix* at the MATLAB prompt and use the resulting plots to answer the questions below.

Q2.6 For which l -values are the phase shifts significantly different from zero?

Q2.7 Which property of the potential makes $S_l \rightarrow 1$ for large l ?

Q2.8 How do you explain that S_l is never on the unit circle, except when $S_l = 1$ for large l ? How would you change the potential in order to get an 'elastic window' where $|S_l| = 1$ before S_l reaches 1.

2.5. Differential cross section

From the partial wave S-matrix, the scattering amplitude $f(\theta)$ follows directly from Eqn 7.6.17:

$$f(\theta) = \frac{1}{2ik} \sum_{l=0}^{\infty} (2l+1)(S_l - 1)P_l(\cos \theta), \quad (12)$$

and the differential cross section is obtained from Eqn 7.1.36

$$\frac{d\sigma}{d\Omega} = |f(\theta)|^2. \quad (13)$$

As will be seen in the numerical evaluation, the differential cross section may vary over many orders of magnitude as a result of constructive and destructive interferences in the partial wave expansion. In cases like this, an accurate evaluation of S_l and the Legendre polynomials is essential.

The next step is to evaluate the differential cross section through Eqn (13) and compare with the measured data points. A template for the MATLAB code for finding $\frac{d\sigma}{d\Omega}$ is provided as `M4_crossection_template.m`. Create your own m-function `M4_crossection.m` in your work directory, include the template code and supply the missing steps specific for the evaluation of f and $\frac{d\sigma}{d\Omega}$.

Again some help concerning the missing code is available in the Appendix.

MATLAB	code for <i>M4_crossection_template.m</i>	MATLAB comments
function	M4_crossection	
load	sparning	
figure		
alphaC12data		
theta	= linspace(0,pi,721);	
f	= zeros(size(theta));	
for index = 1:lmax+1		% add partial waves
l	= index - 1;	
myleg	= legendre(l,cos(theta));	
f	= f + ADD MISSING CODE	% from Eqn (12)
dsigma	= ADD MISSING CODE	% mb/str;from Eqn (13)
%plotting		
semilogy(Data(:,1),Data(:,2), '.')	, ...	% experimental
'MarkerEdgeColor',[1 0 0]);	hold on	% points
semilogy(theta*180/pi,dsigma);		
title(sprintf('including l<=%3.0f',l))		
pause(0.5);	hold off	
end		
title('Differential cross section in mb/str ...	as fcn of angle. Log scale!');	

Start your evaluation of the cross section by entering *M4_crossection* at the MATLAB prompt.

Q2.9 What could be the origin of the oscillatory behaviour at large angles?

The parameters of the Wood-Saxon potential have been adjusted to achieve a good fit to the experimental data (recall, however, that effects of the Coulomb potential are not included in the calculation). To investigate the effect of changing the parameters in the potential go back to your *M2_Schrodinger.m* file and change one of them, say *RV*, by 20%. Re-run the files M2-M4 and look at the final result.

Q2.10 What happens to the fit after the parameter change?

3. Explore quantum scattering using *qmech*

IMPORTANT:

**Before starting this part make sure you have answered all questions above.
Contact your lab. assistant to have them corrected before proceeding.**

In the package *qmech* the solution to a scattering problem in 3D is worked out and visualized using the facilities of MATLAB.

1. In whatever window of *qmech* you are, push 'Return' until you come to the Scat3Dmenu. Verify that the potential is to your liking (i.e. not wildly different from the default potential).

2. Alternatively, restart *qmech* from the MATLAB prompt as in 1.1.. Feel free to change the default Woods-Saxon parameters.

Warning: restarting *qmech* from the MATLAB prompt will clear all variables from your workspace.

3. In the window Scat3Dmenu, push the button 'Parameters'. The scattering problem will be solved for a range of energies. For instance, chose $E_{min}=100$, $E_{max}=160$ and $N=21$. If you have changed the potential, you may need to reconsider the default values for the other parameters. A step size in r at 0.1 fm should be OK for potentials with range 2-3 fm. Make sure to include a sufficient number of partial waves. Push 'Return'.

4. Push 'Evaluation', and then 'Go', for the computer to solve the radial Schrödinger equation for as many energies and partial waves as you have specified. All solutions are evaluated in one shot. You can see how the calculation progresses from $r=0$ to your chosen r_{max} . If your r_{max} was big enough, all curves flatten out. If not, go back and change your parameters. (The graph shows, as a function of r , the phase shift corresponding to a potential that coincides with the true potential up to the radius r , but which equals zero outside this radius. When the curves flatten out, all contributions to the phase shift have been captured.)

5. Push 'Return' and then 'Results'. There are many types of results to look at.

6. First look at the partial wave S-matrix in the complex plane. The curves show the energy variation of $S_l(E)$ for the various l -values, with the highest energy at the endpoint 'L='. Note that all curves fall within the unit circle (unless you have a positive imaginary part in the Woods-Saxon potential; if this is the case you don't have absorption but rather particle creation in the nucleus!).

7. Proceed to look at the corresponding phase shifts as functions of energy (in 2.4 you saw a phase shift at a given energy, as function of l).

8. Look at the inelasticity. High partial waves correspond to large impact parameters, i.e. the particle passes through the tail of the potential and the scattering and absorption is weak.

9. Push 'Return' and 'Differential cross section'. First look at the cross section as a function of scattering angle (in the center of mass system). Unless your energy is very low, you should see a drastic drop from forward to backward angles (logarithmic scale!).

Push 'as a fcn of mom transfer' to see the same curves as functions of momentum transfer (i.e. of $|\vec{k}' - \vec{k}| = \sqrt{k'^2 - 2k^2 \cos \theta + k^2} = k \sin(\theta/2)$) rather than scattering angle. Typically, the differential cross section shows less energy variation when plotted against momentum transfer (much of the energy variation is absorbed through the energy dependence of k !).

10. Push '3Dplot' and you will see the same curves in a different manner.

11. Narrow the energy range to a single value ($E=133$), and push 'as fcn of angle' again. In order to see how the differential cross section is built from more and more partial waves, push 'L-dependence'. Do you have a sufficient number of partial waves for a converged result? If not, go back and modify your parameters, and recalculate the solution. How many partial waves do you need?

12. If you have time, you can add the data points by going to the MATLAB command window and type
`hold on; alphaC12data; semilogy(Data(:,1),Data(:,2),'.'); hold off`

13. Push 'Return' and proceed to '3D Wave functions'. In this window, a single energy has been chosen (the mid-point of your chosen interval). Wave functions are evaluated for points in the x-z plane, and can be visualized in three different ways (the green pop-up menu; try them all).

14. In 'Real(psi)', the real part of the full wave function is shown (without scattering, this would just be the real part of the incoming plane wave, i.e. $\cos(kz)$). In 'Real(psi)-movie' the time-dependence has been introduced to make the visualization more suggestive.

15. In 'Real(psi-scatt)', the plane wave part of the solution has been removed, and only the scattered part of the full wave function is shown.

16. In 'psi^2' the modulus squared of the full wave function is shown, i.e. the probability density. Note that for a strongly absorbing potential, the probability density inside the nucleus is expected to be small. Is this what is shown in your figure? If not, what could be the origin of density enhancements inside or on the back side of the nucleus?

17. If your r-max was somewhat low, you might get more instructive graphs if you go back to 'Parameters' and increase r-max (and recalculate the solution).

18. In 'wavepacket movie', the incoming wave is no longer a plane wave but a wave packet (in the z direction).

This movie requires good computer resources. For instance, if you have chosen too many grid points in the radial variable (too big rmax and/or too small rstep) you may be in trouble here.

19. Repeat steps 14-18 but for a much lower energy (wavelength larger than diameter of scattering region).

Have fun.

Appendix

MATLAB code for <i>legendre_1.m</i>	MATLAB comments
<pre> clear x = linspace(eps,12*pi,101)'; sum(:,1) = cos(x); sum(:,2) = zeros(size(x)); figure for n=0:25 sum(:,2) = sum(:,2) + ... (-1)^n*(4*n+1)*sphbesselj(2*n,x); plot(x,sum); title(sprintf('n <= %3.0f',n)) xlabel('Push any key to continue') pause end clear; close </pre>	<pre> % Note the final ' % LHS % to contain RHS % plots both LHS % and RHS </pre>

MATLAB code for <i>legendre_2.m</i>	MATLAB comments
<pre> clear theta = linspace(0,pi,181)'; z = cos(theta); sum(:,1) = cos(15*z); sum(:,2) = zeros(size(z)); figure for n = 0:20 myleg = legendre(2*n,z); sum(:,2) = sum(:,2)+(-1)^n*(4*n+1)*... sphbesselj(2*n,15)*myleg(1,:)'; felet = (sum(91,2)-sum(91,1))/sum(91,1); plot(z,sum) title([sprintf('n <= %3.0f',n) ': ' ... sprintf('rel error at 90 deg = %1.3e',felet)]); xlabel('Push any key to continue') pause end clear; close </pre>	<pre> % Note the final ' % LHS % to contain RHS % plots both LHS % and RHS </pre>

Hints for the code *M3_Smatrix*:

The evaluation of S_l is to be done for many partial waves (i.e. l -values), and can be executed in an explicit loop over l , like in *M2_Schrodinger*. This, however, is contrary to the spirit of MATLAB, and instead the matrix handling power of MATLAB will be exploited. Therefore, L has been introduced as the (row) vector of all l -values, and since $hplus$ is evaluated using L , $hplus$ is also obtained as a (row) vector with elements $h_l^+(kR)$ for $l = 0, 1, \dots, lmax$. In the same spirit, $beta$ will be a vector of a similar kind. It can be directly obtained according to Eqn (10) using the two vectors $uend(:, 1)$ and $uend(:, 2)$ calculated in *M2_Schrodinger* (and saved in *sparing*). For this purpose the MATLAB element by element division $./$ is needed (recall that if a has elements a_1, a_2, \dots and b has elements b_1, b_2, \dots then $a./b$ has elements $a_1/b_1, a_2/b_2, \dots$; also available is element by element multiplication $.*$ and exponentiation $.^$). Note that $uend(:, 1)$ and $uend(:, 2)$ are column vectors, and that e.g. $uend(:, 1)'$ is the corresponding row vector, but with complex conjugated elements. What is needed here is therefore the (row) vector $conj(uend(:, 1)')$, etc.

Similarly, the vector Sl is obtained according to Eqn (11) using the vectors $beta$, $hplus$ and $hplusprim$. Note that $h_l^-(kR) = (h_l^+(kR))^*$ or, translated into MATLAB code, $hminus = conj(hplus)$.

The phase shift δ_l is obtained (element by element) from Sl using the MATLAB function *angle*.

The *unwrap* and *fliplr* in the plotting part serve to make δ_l vary in a smooth manner with l (i.e. prevent jumps of 2π), and to make δ_l go to 0 (rather than, e.g. to -2π) for $l \rightarrow \infty$.

Hints for the code *M4_crosssection*:

The MATLAB function *legendre(L, theta)* allows $theta$ but not L to be a vector. In the template, an explicit loop over l is therefore suggested. Note that *legendre(l, theta)* returns both ordinary and associated Legendre function, with ordinary Legendre functions in the first row. What will be needed in the calculation of f is therefore *myleg(1, :)*, which is a row vector as long as $theta$.

Recall from *M3_Smatrix* that the element $Sl(index)$ corresponds to the partial wave $l = index - 1$.

For comparison with the experimental points, the cross section must be expressed in mb/str. Recall that 1 barn = 100 fm².