© 2007 WILEY-VCH Verlag GmbH & Co

30
Quantum Scattering via Integral Equations

In this chapter we develop techniques to solve the singular, inhomogeneous integral equation appropriate for quantum scattering, a different and more advanced problem than that in Chap. 29. After these two chapters we hope the reader views both integral and differential equations as soluble.

Problem: The problem is essentially the same problem of a particle interacting with a nonlocal potential discussed in Chap. 29 (Fig. 29.1), only now we need to deduce the scattering (Fig. 30.1) that occurs when a particle passes through a dense medium.

30.1
Lippmann–Schwinger Equation (Theory)

Because scattering experiments measure scattering amplitudes, it is convenient to convert the Schrödinger equation into an equation dealing with amplitudes rather than wave functions. An integral form of the Schrödinger equation dealing with the amplitude R (reaction matrix) is the *Lippmann–Schwinger equation*:[1]

$$R(k',k) = V(k',k) + \frac{2}{\pi}\mathcal{P}\int_0^\infty dp \frac{p^2 V(k',p)R(p,k)}{(k_0^2 - p^2)/2\mu} \tag{30.1}$$

Note that Eq. (30.1) requires more than just an integral to evaluate. It is an integral equation in which $R(p,k)$ is integrated over, yet since $R(p,k)$ is unknown, the integral cannot be evaluated until after the equation is solved!

The symbol $\mathcal{P}$ in (30.1) indicates the Cauchy principal-value prescription for avoiding the singularity arising from the zero of the denominator (we discuss how to do that in Section 30.1.1). This equation describes the scattering of two particles with reduced mass and center-of-mass energy (Fig. 30.1):

$$\mu = \frac{m_1 m_2}{m_1 + m_2} \qquad E = \frac{k_0^2}{2\mu} \tag{30.2}$$

[1] To keep the presentation simple, our equations are given in the partial-wave basis but without the l subscripts to indicate it.

Computationyal Physics. Problem Solving with Computers (2nd edn).
Rubin H. Landau, Manuel José Páez, Cristian C. Bordeianu
Copyright © 2007 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
ISBN: 978-3-527-40626-5

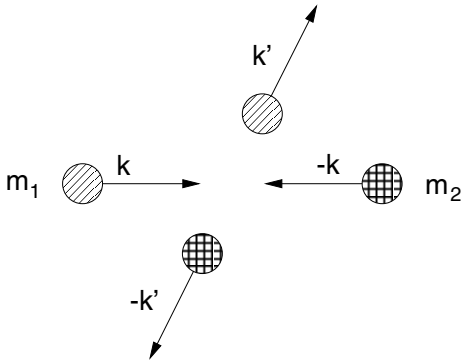

Fig. 30.1 The scattering of mass m_1 and m_2 in their center-of-momentum system.

and initial and final center-of-mass momenta k and k'. The diagonal matrix element $R(k_0, k_0)$ is the experimental scattering amplitude needed to solve your **problem**.

30.1.1 Singular Integrals (Mathematics)

A *singular* integral

$$\mathcal{G} = \int_a^b g(k)dk \tag{30.3}$$

is one in which the integrand $g(k)$ is singular at a point k_0 within the interval $[a, b]$, and yet the integral $\mathcal{G}$ is finite. (If the integral itself was infinite, we could not compute it.) Unfortunately, computers are notoriously bad at dealing with infinite numbers, and if an integration point gets too near to the singularity, severe subtractive cancellation or overflow occurs. Consequently, we apply some results from complex analysis before evaluating singular integrals numerically.[2]

In Fig. 30.2 we show three ways in which the singularity of an integrand can be avoided. The paths in A and B move the singularity slightly off the real k axis by giving the singularity a small imaginary part $\pm i\epsilon$. The Cauchy principal-value prescription $\mathcal{P}$ (Fig. 30.2 right) is seen to "pinch" both sides of the singularity at k_0, but not to pass through it:

$$\mathcal{P}\int_{-\infty}^{+\infty} f(k)dk = \lim_{\epsilon\to 0}\left[\int_{-\infty}^{k_0-\epsilon} f(k)dk + \int_{k_0+\epsilon}^{+\infty} f(k)dk\right] \tag{30.4}$$

[2] Ref. [88] describes a different approach using *Maple* and *Mathematica*.

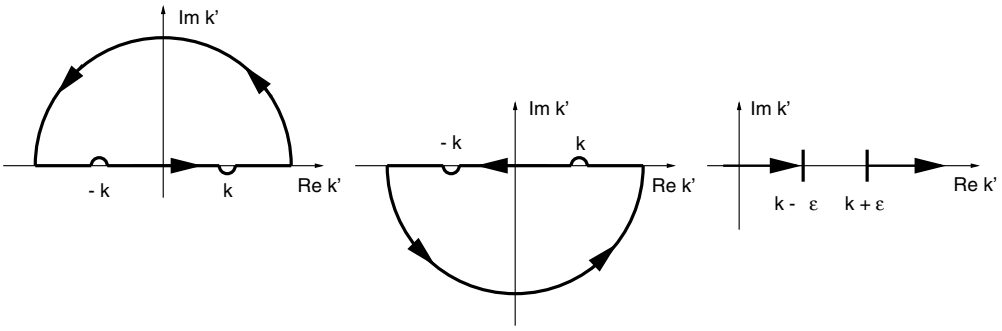

Fig. 30.2 Three different paths in the complex k' plane used to evaluate line integrals when there are singularities. Here the singularities are at k and $-k$, and the integration variable is k'.

The preceding three prescription are related by

$$\int_{-\infty}^{+\infty} \frac{f(k)dk}{k-k_0 \pm i\epsilon} = \mathcal{P}\int_{-\infty}^{+\infty} \frac{f(k)dk'}{k-k_0} \mp i\pi f(k_0) \tag{30.5}$$

which follows from Cauchy's residue theorem and some contour distortions.

30.1.2 Numerical Principal Values

A numerical principal value limit (30.4) is awkward because computers have limited precision. A better prescription for computers follows from the calculus relation

$$\mathcal{P}\int_{-\infty}^{+\infty} \frac{dk}{k-k_0} = 0 \tag{30.6}$$

This equation means that the curve of $1/(k-k_0)$ as a function of k has equal and opposite areas on both sides of the singular point k_0. If we break the integral up into one over positive k and one over negative k, a change of variable $k \to -k$ permits us to express (30.6) as

$$\mathcal{P}\int_{0}^{+\infty} \frac{dk}{k^2-k_0^2} = 0 \tag{30.7}$$

We observe that the principal-value exclusion of the singular point's contribution is equivalent to a simple subtraction of the zero integral (30.7):

$$\mathcal{P}\int_{0}^{+\infty} \frac{f(k)dk}{k^2-k_0^2} = \int_{0}^{+\infty} \frac{[f(k)-f(k_0)]dk}{k^2-k_0^2} \tag{30.8}$$

We notice that there is no $\mathcal{P}$ on the RHS of (30.8) because the integrand is no longer singular at $k=k_0$ (it is proportional to the df/dk) and can therefore be

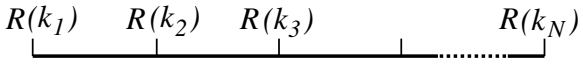

Fig. 30.3 The grid in momentum space on which the integral equation for the R is solved.

evaluated numerically as can any other integral! The integral (30.8) is called the *Hilbert transform* of f and also arises in inverse problems.

30.1.3 Reducing Integral to Matrix Equations (Method)

Now that we know how to handle singular integrals, we return to our problem of a singular integral equation. We want to reduce the integral equation into a set of linear equations that are then solved with matrix operations. We need to solve the integral equation (30.1) with the potential (30.22). The momentum[3] k_0 is related to the energy E and the reduced mass μ by (30.2). The experimental observable that results from a solution of (30.1) is the amplitude $R(k_0, k_0)$, or equivalently, the scattering phase shift δ_l:

$$R(k_0, k_0) = -\frac{\tan \delta_l}{\rho} \qquad \rho = 2\mu k_0 \tag{30.9}$$

The procedure for the computer solution of (30.1) uses (30.8) to rewrite the principal-value prescription as a definite integral [89]:

$$R(k', k) = V(k', k) + \frac{2}{\pi} \int_0^\infty dp \frac{p^2 V(k', p) R(p, k) - k_0^2 V(k', k_0) R(k_0, k)}{(k_0^2 - p^2)/2\mu} \tag{30.10}$$

We convert this integral equation into linear equations by approximating the integral as a sum over N integration points (usually Gauss quadrature) $\{k_j;\ j = 1, N\}$ with weights w_j:

$$\begin{aligned} R(k, k_0) \ \simeq\ & V(k, k_0) + \frac{2}{\pi} \sum_{j=1}^{N} \frac{k_j^2 V(k, k_j) R(k_j, k_0) w_j}{(k_0^2 - k_j^2)/2\mu} \\ & - \frac{2}{\pi} k_0^2 V(k, k_0) R(k_0, k_0) \sum_{m=1}^{N} \frac{w_m}{(k_0^2 - k_m^2)/2\mu} \end{aligned} \tag{30.11}$$

We note that the last term in (30.11) implements the principal-value prescription and cancels the singular behavior of the previous term.

Equation (30.11) contains the $N + 1$ unknowns $R(k_j, k_0)$ for $j = 1, N$, and $R(k_0, k_0)$. We turn it into $N + 1$ simultaneous equations by evaluating it for

[3] We are formulating this problem with "natural" units in which Planck's constant $\hbar \equiv 1$. This means that there is no difference between momentum and wave vectors.

$N+1$ k values on a grid consisting of the observable momentum k_0 and the integration points (Fig. 30.3):

$$k = k_i = \begin{cases} k_j, & j = 1, N \quad \text{(quadrature points)}, \\ k_0, & i = 0 \quad\quad \text{(observable point)}. \end{cases} \tag{30.12}$$

There are now $N+1$ linear equations for $N+1$ unknowns $R_i \stackrel{\text{def}}{=} R(k_i, k_0)$:

$$R_i \;=\; V_i \;+\; \frac{2}{\pi} \sum_{j=1}^{N} \frac{k_j^2 V_{ij} R_j w_j}{(k_0^2 - k_j^2)/2\mu} \;-\; \frac{2}{\pi} k_0^2 V_{ii} R_0 \sum_{m=1}^{N} \frac{w_m}{(k_0^2 - k_m^2)/2\mu} \tag{30.13}$$

We express these equations in the matrix form $\mathbf{Ax} = \mathbf{b}$ by combining the denominators and weights into a single denominator vector D:

$$D_i = \begin{cases} +\frac{2}{\pi} \frac{w_i k_i^2}{(k_0^2 - k_i^2)/2\mu} & \text{for } i = 1, N, \\ -\frac{2}{\pi} \sum_{j=1}^{N} \frac{w_j k_0^2}{(k_0^2 - k_j^2)/2\mu} & \text{for } i = N+1 \end{cases} \tag{30.14}$$

The linear equations (30.13) now assume that the matrix form

$$R - DVR \;=\; [1 - DV]\, R \;=\; V \tag{30.15}$$

where R and V are *column vectors* of length $N_1 \equiv N+1$:

$$[R] \;=\; \begin{pmatrix} R_{1,N_1} \\ R_{2,N_1} \\ \ddots \\ R_{N_1,N_1} \end{pmatrix} \qquad\qquad [V] \;=\; \begin{pmatrix} V_{1,N_1} \\ V_{2,N_1} \\ \ddots \\ V_{N_1,N_1} \end{pmatrix} \tag{30.16}$$

We call the matrix $[1 - Dv]$ in (30.15) the wave matrix F, and write the integral equation as the matrix equation:

$$[F][R] \;=\; [V] \qquad\qquad F_{ij} \;=\; \delta_{ij} - D_j V_{ij} \tag{30.17}$$

With R the unknown vector, (30.17) is in the standard form $AX = B$, which can be solved by the mathematical subroutine libraries discussed in Chap. 8.

30.1.4 Solution via Inversion, Elimination

An elegant (but alas not efficient) solution to (30.17) is by matrix inversion:

$$[R] \;=\; [F]^{-1}[V] \tag{30.18}$$

Because the inversion of even complex matrices is a standard routine in mathematical libraries, (30.18) is a *direct solution* for the R matrix. A more efficient

approach is to find an $[R]$ that solves $[F][R] = [V]$ without computing the inverse. This is accomplished by Gaussian *elimination*.

30.1.5 Solving $i\epsilon$ Integral Equations ⊙

The integral equation most commonly encountered in quantum mechanics corresponds to outgoing wave boundary conditions. This means that the singularity is handled by giving the energy $k_0^2/2\mu$ a small positive imaginary part $i\epsilon$. This procedure leads to the Lippmann–Schwinger equation for the T matrix:

$$T(k',k) \;=\; V(k',k) \;+\; \frac{2}{\pi}\int_0^\infty dp\,\frac{p^2 V(k',p)T(p,k)}{(k_0^2-p^2+i\epsilon)/2\mu} \tag{30.19}$$

Solving this equation is essentially the same as solving (30.1) for the $R(k',k)$ matrix. We use the identity (30.5) and decompose the $i\epsilon$ integral into a principal-value part and an on-shell term:

$$T(k',k) = V(k',k) + \frac{2}{\pi}\mathcal{P}\int_0^\infty dp\,\frac{p^2 V(k',p)T(p,k)}{(k_0^2-p^2)/2\mu} - 2i\mu k_0 V(k',k_0)T(k_0,k).$$

Now the last term is incorporated into the numerical analysis by adding an imaginary term to the D matrix (30.14):

$$D_{N+1} \;=\; -\frac{2}{\pi}\sum_{j=1}^{N}\frac{w_i k_0^2}{(k_0^2-k_j^2)/2\mu} \;-\; 2\mu i k_0 \tag{30.20}$$

The solution proceeds as before, only now with complex matrices arising from the new definition of D. The resulting on-shell T matrix element is related to the same experimental phase shift as before, only now through the complex expression

$$T(k_0,k_o) = -\frac{e^{i\delta_l}\sin\delta_l}{\rho} \qquad\qquad \rho = 2\mu k_0 \tag{30.21}$$

30.1.6 Delta-Shell Potential Implementation

In Section 29.1.2 we discussed the *delta-shell potential* and gave its momentum-space matrix element for $l = 0$ partial waves:

$$V(r) \;=\; \frac{\lambda}{2\mu}\delta(r-b) \tag{30.22}$$

$$V(k',k) \;=\; \frac{\lambda}{2\mu k' k}\sin(k'b)\sin(kb) \tag{30.23}$$

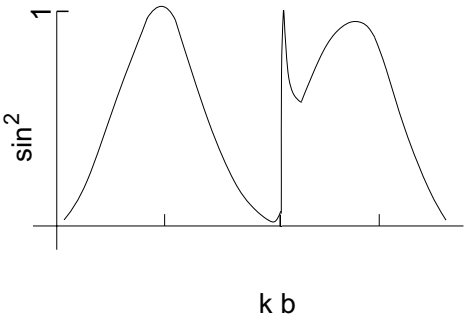

Fig. 30.4 The energy dependence of that part of the scattering cross section arising from the $l = 0$ phase shift.

This is one of the few potentials for which the Lippmann–Schwinger integral equation (30.1) has an analytic solution [8]:

$$e^{i\delta} \sin\delta = \frac{-\lambda k_0 b^2 \sin(k_0 b)}{1 + i\lambda k_0 b^2 \sin(k_0 b)\,[\cos(k_0 b) + i \sin(k_0 b)]} \tag{30.24}$$

With these equations we can calculate the $l = 0$ phase shift and compare it to that obtained from your numerical solution of the integral Schrödinger equation. In Fig. 30.4 we give a plot of $\sin^2\delta_0$ versus kb. This is proportional to the scattering cross section arising from the $l = 0$ phase shift. It is seen to reach its maximum value at energies corresponding to resonances. Your numerical results should be similar to this, although it may be difficult to reproduce the very sharp energy dependence.

1. Set up the matrices $V(i,j)$, $V(i)$, $D(j)$, and $F(i,j)$ according to (30.16)-(30.23). Use Gaussian quadrature points with at least $N = 16$ for your grid.
2. Employ a matrix inversion routine you have obtained from a library to calculate F^{-1}.
3. Calculate the vector R by matrix multiplication $R = F^{-1}V$.
4. Deduce the S-wave phase shift δ from your R vector:

$$R(k_0, k_0) = R_{N1,N1} = -\frac{\tan\delta_l}{\rho} \qquad \rho = 2\mu k_0 \tag{30.25}$$

5. Estimate the precision of your solution by increasing the number of grid point in steps of 4. If your phase shift changes in the second or third decimal place, you probably have that much precision.

6. Plot $\sin^2 \delta_0$ versus energy $E = k_0^2/2\mu$ starting at zero energy and ending at energies where the phase shift is again small. Your results should be similar to those in Fig. 30.4 (calculated from the analytic result). Note that a *resonance* occurs when δ_l increases rapidly through $\pi/2$; that is, when $\sin^2 \delta_0 = 1$.

7. Check your answer against the analytic results (30.24).

30.1.7 Scattering Wave Function (Exploration)

1. The F^{-1} matrix that occurred in our solution to the integral equation,

$$R = F^{-1}V = (1 - VG)^{-1}V \tag{30.26}$$

is actually quite useful. In scattering theory it is known as the *wave matrix* because it is used in the expansion of the wave function:

$$u_l(r) = N_0 \sum_{i=1}^{N} j_l(k_i r) F(k_i, k_0)^{-1} \tag{30.27}$$

Here N_0 is a normalization constant and standing-wave boundary conditions are built into u_l if the R matrix is used to calculate F. Plot this wave function and compare it to a free wave.

2. Solve for the part of the scattering cross section arising from the $l = 1$ phase shift for $0 \leq kb \leq 2\pi$.

Listing 30.1: `Scatt.java` solves the Lippmann–Schwinger integral equation for scattering from a delta-shell potential. The singular integral equations are regularized by a subtraction, converted to matrix equations using Gaussian grid points, and then solved with JAMA.

```
// Scatt.java: Soln of Lippmann-Schwinger in p space for scattering

import Jama.*;
import java.io.*;
import java.util.*;

public class Scatt {
  public static void main(String[] argv) throws IOException,
                                                 FileNotFoundException {
    PrintWriter q = new PrintWriter(
                            new FileOutputStream("sin2.dat"), true);
    int n, i, j, m, Row, Column, M = 300;
    double pot, lambda, scale, ko, Temp, shift, shiftan, sin2, k2;
    double pi = 3.141592653589793238 46, b = 10., RN1, potlast=0.0;
    double[][] F = new double[M][M]; double[] k = new double[M];
    double[] w = new double[M]; double[]D = new double[M];
    double[] r =new double[M]; double[] V = new double[M];
    double[][]P = new double[M][M]; double[][]L = new double[M][M];
    double[][]U = new double[M][M];
    n = 26;                scale = n/2;          pot = 0. ;
    shiftan = 0.;       lambda =1.5;            // Set up Gauss points
    Gauss.gauss(n, 2, 0., scale, k, w);
    ko = 0.02;
    for ( m=1;m<901;m++)  {                          // Set up D matrix
      k[n] = ko;
      for ( i=0; i<= n-1; i++ ){
      D[i]=2/pi*w[i]*k[i]*k[i]/(k[i]*k[i]-ko*ko);
      }
      D[n] = 0. ;
      for (j=0; j <= n-1;j++) D[n]=D[n]+w[j]*ko*ko/(k[j]*k[j]-ko*ko);
      D[n] = D[n]*(-2./pi);
      for ( i=0; i <= n; i++ ) {       // Set up F matrix and V vector
        for ( j=0; j <= n; j++ )  {
          pot = -b*b * lambda * Math.sin(b*k[i])
                                  * Math.sin(b*k[j])/(k[i]*b*k[j]*b);
          F[i][j] = pot*D[j];
          if (i==j) F[i][j] = F[i][j] + 1.;
        }
        V[i] = pot;
      }                                  // Change arrays into matrices
      Matrix Fmat = new Matrix(F, n+1, n+1);
      Matrix Vvec = new Matrix( n+1, 1);
      Matrix Finv = Fmat.inverse();
      for ( i=0; i <= n; i++ )  Vvec.set(i, 0, V[i]);
      Matrix R = Finv.times(Vvec);                    // Invert matrix
      RN1 = R.get(n, 0);                          // Get last value of R
                                                   // Define phase shift
      shift = Math.atan(-RN1*ko);
      sin2 = Math.sin(shift)*Math.sin(shift);
      q.println(ko*b + "  " + sin2);
      ko=ko+0.2*3.141592/1000.0;
    }
    System.out.println("Output in sin2.dat");
   }
}
```