29
Quantum Bound States via Integral Equations

The power and accessibility of high-speed computers has changed the view as to which theories and equations in physics are soluble. In Chaps. 15 and 19 we saw how even nonlinear differential equations can be easily solved to give new insight into the physical world. In this chapter we examine how integral equations can be solved for both bound and scattering quantum states. This chapter extends our treatment of the eigenvalue problem, solved as a coordinate (r) space differential equation in Chap. 15, to the equivalent problem solved as a homogeneous integral equation in momentum (p) space. Chap. 30 examines quantum scattering in momentum space, a more advanced problem than bound states. After these chapters we hope the reader views both integral and differential equations as soluble.

Problem: A projectile particle interacts with the particles in a medium through which the projectile passes (Fig. 29.1). The multiple-particle nature of the interaction leads to a nonlocal potential in which the potential at $\mathbf{r}$ depends on the wave function at all $\mathbf{r'}$ values. This changes the interaction term in the Schrödinger equation to [30]:

$$V(r)\psi(r) \rightarrow \int dr' V(r,r')\psi(r') \tag{29.1}$$

The integration in (29) leads to a Schrödinger equation that is a combined integral and differential ("integrodifferential") equation:

$$-\frac{1}{2\mu}\frac{d^2\psi(r)}{dr^2} + \int dr' V(r,r')\psi(r') = E\psi(r) \tag{29.2}$$

Your **problem** is to figure out how to find the bound-state energies E_n and wave functions ψ_n for the integral equation in (29.2).[1]

[1] We use "natural" units in which Planck's constant $\hbar \equiv 1$, and so there is no difference between momenta and wave vectors.

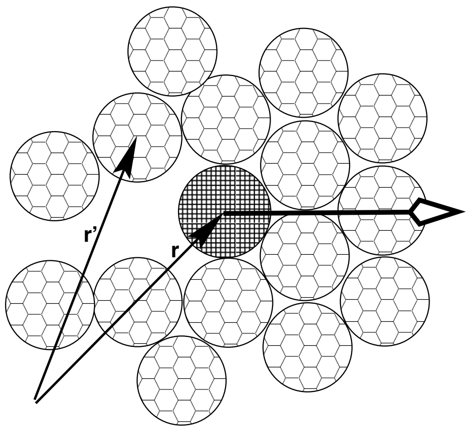

Fig. 29.1 A particle (dark) moving to the right through a medium with multiple particles present. The nonlocality of the potential felt by the dark particle at **r** arises from the particle interactions at all **r'** with the other particles.

29.1
k-Space Schrödinger Equation (Theory)

One way of dealing with Eq. (29.2) is by going to momentum space and the partial-wave basis, in which it becomes the integral equation [30]:

$$\frac{k^2}{2\mu}\psi_n(k) + \frac{2}{\pi}\int_0^\infty dpp^2 V(k,p)\psi_n(p) = E_n\psi_n(k) \tag{29.3}$$

Here $V(k,p)$ is the momentum–space representation (Fourier transform) of the potential, $\psi_n(k)$ is the momentum–space wave function (the probability amplitude for finding the particle with momentum k) and the subscript n distinguishes solutions of different energies E_n. To be more specific, $\psi_n(k)$ is converted to $\psi_n(r)$ by a Bessel transform,

$$\psi_n(r) = \int_0^\infty dk\psi_n(k)j_l(kr)k^2 \tag{29.4}$$

and the momentum–space potential $V(k',k)$ is obtained from the $V(r',r)$ by a double Bessel transform:

$$V(k,p) \;=\; (2\pi)^3\int_0^\infty drr^2 j_l(kr)V(r)j_l(pr) \qquad (\text{local } V). \tag{29.5}$$

The $j_l(kr)$'s are spherical Bessel functions,

$$j_0(z) = \frac{\sin z}{z} \qquad\qquad j_1(z) = \frac{\sin z}{z^2} - \frac{\cos z}{z} \qquad\qquad \dots \tag{29.6}$$

for which, you may recall, we developed a technique for computing in Chap. 3. For the $l = 0$ case considered in this chapter, the potential matrix element is

$$V(k,p) \;=\; \frac{(2\pi)^3}{kp}\int_0^\infty drr^2\sin(kr)V(r)\sin(pr) \tag{29.7}$$

Equation (29.3) is an integral equation for $\psi_n(k)$. It is more than an integral expression for $\psi_n(k)$ because the integral in it cannot be evaluated until $\psi_n(p)$ is known for all p's; that is, until the equation is solved! Nevertheless, we shall see that we can transform this problem into a matrix eigenvalue problem. The matrix eigenvalue problem is easy to solve on the computer using a mathematical subroutine library.[2].

Those with a mathematical bent may care to observe that the standard formulation of the bound-state problem imposes the boundary condition that the wave function decays exponentially as $r \to \infty$. While the formulation given in this section does not explicitly impose that condition, it assumes that the wave packet is normalizable. The two conditions are equivalent because only an exponentially decaying wave function will be normalizable.

29.1.1 Integral to Linear Equations (Method)

One technique for solving integral equations is to transform them to linear equations that can be solved as matrix equations. This is also done in our discussion of quantum scattering in Section 30.1.3. We approximate the integral over the potential as a weighted sum over N integration points (usually Gauss quadrature[3]), $p = k_j,\ j = 1, N$:

$$\int_0^\infty dpp^2V(k,p)\psi_n(p) \;\simeq\; \sum_{j=1}^{N} w_jk_j^2V(k,k_j)\psi_n(k_j). \tag{29.8}$$

This converts the integral equation (29.3) into the algebraic equation

$$\psi_n(k)\frac{k^2}{2\mu}\psi_n(k) \;+\; \frac{2}{\pi}\sum_{j=1}^{N} w_jk_j^2V(k,k_j)\psi_n(k_j) \;=\; E_n \tag{29.9}$$

For a given value of the label n, (29.9) contains the N unknowns $\psi_n(k_j)$, the single unknown E_n, and the unknown function $\psi_n(k)$. We eliminate the unknown function $\psi_n(k)$ by evaluating the equation on a grid (Fig. 29.2) composed of the same N k_i values used to approximate the integral. This leads to the set of N coupled linear equations in $N+1$ unknowns:

$$\frac{k_i^2}{2\mu}\psi_n(k_i) + \frac{2}{\pi}\sum_{j=1}^{N} w_jk_j^2\ V(k_i,k_j)\psi_n(k_j) = E_n\psi_n(k_i) \qquad i = 1, N \tag{29.10}$$

[2] The use of libraries is described in Chap. 8
[3] See Chap. 5 for a discussion of numerical integration.

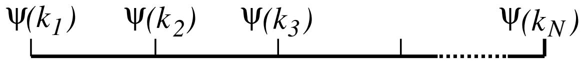

Fig. 29.2 The grid in momentum space on which the integral equation for the wave function is solved.

As a case in point, if $N = 2$ we would have the two simultaneous linear equations

$$\begin{aligned}
\frac{k_1^2}{2\mu}\psi_n(k_1) + \frac{2}{\pi} w_1 k_1^2\, V(k_1,k_1)\psi_n(k_1) + w_2 k_2^2\, V(k_1,k_2) &= E_n\psi_n(k_1) \\
\frac{k_2^2}{2\mu}\psi_n(k_2) + \frac{2}{\pi} w_1 k_1^2\, V(k_2,k_1)\psi_n(k_1) + w_2 k_2^2\, V(k_2,k_2)\psi_n(k_2) &= E_n\psi_n(k_2)
\end{aligned}$$

We write our coupled dynamical equations in the matrix form as

$$[H][\psi_n] = E_n[\psi_n] \tag{29.11}$$

or as explicit matrices

$$\begin{pmatrix}
\frac{k_1^2}{2\mu} + \frac{2}{\pi}V(k_1,k_1)k_1^2 w_1 & \frac{2}{\pi}V(k_1,k_2)k_2^2 w_2 & \cdots & \frac{2}{\pi}V(k_1,k_N)k_N^2 w_N \\
\frac{2}{\pi}V(k_2,k_1)k_1^2 w_1 & \frac{2}{\pi}V(k_2,k_2)k_2^2 w_2 + \frac{k_2^2}{2\mu} & \cdots & \\
\ddots & & & \\
\cdots & \cdots & \cdots & \frac{k_N^2}{2\mu} + \frac{2}{\pi}V(k_N,k_N)k_N^2 w_N
\end{pmatrix}$$

$$\times \begin{pmatrix} \psi_n(k_1) \\ \psi_n(k_2) \\ \ddots \\ \psi_n(k_N) \end{pmatrix} = E_n \begin{pmatrix} \psi_n(k_1) \\ \psi_n(k_2) \\ \ddots \\ \psi_n(k_N) \end{pmatrix} \tag{29.12}$$

Equation (29.11) is the matrix representation of the Schrödinger equation (29.3). The wave function $\psi_n(k)$ on the grid is the $N \times 1$ vector

$$[\psi_n(k_i)] = \begin{pmatrix} \psi_n(k_1) \\ \psi_n(k_2) \\ \ddots \\ \psi_n(k_N) \end{pmatrix} \tag{29.13}$$

where the subscript n on ψ is an energy index.

The acute reader may be questioning the possibility of solving N equations for $N + 1$ unknowns. That reader is wise; only sometimes, and only for certain values of E_n (eigenvalues) will the computer be able to find solutions. To see how this arises, we try to apply the matrix inversion technique (which we will use successfully for scattering in Unit II of this chapter). We rewrite (29.11) as

$$[H - E_n I][\psi_n] = [0] \tag{29.14}$$

We now multiply both sides of this equation by the inverse of $[H - E_n I]$ to obtain the formal solution

$$[\psi_n] = [H - E_n I]^{-1}[0] \tag{29.15}$$

This equation tells us that one of two things is happening. If the inverse exists, then we have the *trivial* solution $\psi_n \equiv 0$, which is not very interesting. Alternatively, if nontrivial solutions are to exist, then our assumption that the inverse exists must be incorrect. Yet we know from the theory of linear equations that the inverse fails to exist when the determinant vanishes:

$$\det[H - E_n I] = 0 \qquad \text{(bound-state condition)} \tag{29.16}$$

Equation (29.16) is the additional equation needed to find unique solutions to the eigenvalue problem. There is, as the case may be, no guarantee that solutions of (29.16) can always be found. When they are found, they correspond to the *eigenvalues* of (29.11) and are the bound-state energies of our physical system.

29.1.2
Delta-Shell Potential (Model)

To keep things simple, and to have an analytic answer to compare with, we consider the local, delta-shell potential:

$$V(r) \; = \; \frac{\lambda}{2\mu}\delta(r-b) \tag{29.17}$$

This would be a good model for an interaction that occurs when two particles are predominantly a fixed distance b apart. Because (29.17) is a local potential, we use (29.5) to determine its momentum–space representation:

$$V(k', k) \; = \; \int_0^\infty r^2 j_l(k'r')\frac{\lambda}{2\mu}\delta(r-b) j_l(kr)dr \; = \frac{\lambda b^2}{2\mu} j_l(k'b) j_l(kb) \tag{29.18}$$

where the subscript l indicates the angular momentum state for which we are solving the problem. (*Beware:* The potential $V(k', k)$ oscillates too much to be well behaved. To obtain stable answers, you need to look at the functional dependence of the integrand, and distribute your integration points on that account.)

If the energy is parameterized in terms of a wave vector κ by $E = -\kappa^2/2\mu$, then for this potential there is, at most, one bound state for each value of the angular momentum l, and it satisfies the transcendental equation [8]

$$1 - \frac{\lambda}{i\kappa}(i\kappa b)^2 j_l(i\kappa b)\left[n_l(i\kappa b) - i j_l(i\kappa b)\right] = 0 \tag{29.19}$$

For $l = 0$, this takes the simple form

$$e^{-2\kappa b} - 1 = \frac{2\kappa}{\lambda} \qquad (l = 0) \tag{29.20}$$

Note, bound states occur only for attractive potentials, and even then, only if the attraction is strong enough. For the present case this means that we must have $\lambda < 0$.

Exercise: Pick some values of b and λ, and use them to verify with a hand calculation that (29.20) can be solved for κ.

29.1.3 Binding Energies Implementation

In List 29.1 we present our solution of the integral equation for bound states of the delta-shell potential using the JAMA matrix library and the `gauss` method for Gaussian quadrature points and weights. An actual computation may follow two paths. The first would call subroutines to evaluate the determinant of the $[H - E_n I]$ matrix in (29.16) and then to *search* for those values of energy for which the computed determinant vanishes. This provides E_n, but not wave functions. The other approach would call an eigenproblem solver that may give some or all eigenvalues and eigenfunctions. In both cases, the solution is obtained iteratively, and you may be required to guess starting values for both the eigenvalues and eigenvectors.

1. Write your own, or modify the code on the CD, so that you can solve the integral equation (29.11) for the delta-shell potential (29.18). You can do this either by evaluating the determinant of $[H - E_n I]$ and then finding the E for which the determinant vanishes, *or* by finding the eigenvalues and eigenvectors for this H.
2. Set the scale by setting $2\mu = 1$ and $b = 10$.
3. Set up the potential and Hamiltonian matrices $V(i, j)$ and $H(i, j)$ for Gaussian quadrature integration with at least $N = 16$ grid points.
4. Adjust the value and sign of λ to find the $l = 0$ bound state. A good approach is to start with a large negative value for λ, and then make it less negative. You should find the eigenvalue moves up in energy.

5. Try increasing the number of grid points in steps of 8, for example; 16, 24, 32, 64, . . ., and see how the energy changes.

6. *Note:* Your eigenenergy solver may return several eigenenergies. Only the true bound state will be at negative energy and will be stable as the number of grid points change.

7. Extract the best value for the bound-state energy and estimate its precision by seeing how it changes with the number of grid points.

8. Check your solution by comparing the RHS and LHS in the matrix multiplication $[H][\psi] = E[\psi]$.

9. Verify that, regardless of the potential's strength, there is only a single bound state.

29.1.4 Wave Function (Exploration)

1. Determine the momentum–space wave function $\psi_n(k)$. Does it fall off at $k \to \infty$? Does it oscillate? Is it well behaved at the origin?

2. Determine the coordinate–space wave function via the Bessel transform

$$\psi_0(r) = \int_0^\infty dk\psi_0(k)j_0(kr)k^2 \tag{29.21}$$

Does $\psi_0(r)$ fall off like you would expect for a bound state? Does it oscillate? Is it well behaved at the origin?

3. Compare the r dependence of this $\psi_0(r)$ to the analytic wave function:

$$\psi_0(r) \propto \begin{cases} e^{-\kappa r} - e^{\kappa r} & \text{for} \quad r < b \\ e^{-\kappa r} & \text{for} \quad r > b \end{cases} \tag{29.22}$$

4. Deduce the energy of the $l = 1$ bound state.

Listing 29.1: `Bound.java` solves the Lippmann–Schwinger integral equation for bound states within a delta-shell potential. The integral equations are converted to matrix equations using Gaussian grid points, and they are solved with JAMA.

```
// Bound.java: Bound states in p space for delta shell potential
//                         uses JAMA and includes Gaussian integration

import Jama.*;

public class Bound {
  static double min =0., max =200., u =0.5, b =10. ;      // Class vars
  public static void main(String[] args) {

    System.out.println("M, lambda, eigenvalue");
    for ( int M = 16; M <= 128; M += 8) {
      for ( int lambda = -1024; lambda < 0; lambda /= 2) {
        double A[][] = new double[M][M],                // Hamiltonian
       WR[] = new double[M], VR,          // RE eigenvalues , potential
        k[] = new double[M], w[] = new double[M];            // Pts & wts
                                              // Call gauss integration
        gauss(M, min, max, k, w);
                                                     // Set Hamiltonian
        for ( int i=0; i < M; i++ )
          for ( int j=0; j < M; j++ ) {
           VR=lambda/2/u*Math.sin(k[i]*b)/k[i]*Math.sin(k[j]*b)/k[j];
           A[i][j] = 2/Math.PI*VR*k[j]*k[j]*w[j];
           if (i == j) A[i][j] += k[i]*k[i]/2/u;
          }
                                           // Eigenvalue decomposition
        EigenvalueDecomposition E
                     = new EigenvalueDecomposition(new Matrix(A));
       WR = E.getRealEigenvalues();                    // RE eigenvalues

        // Matrix V = E.getV();                           // Eigenvectors

        for ( int j=0; j < M; j++ ) if (WR[j] < 0) {
          System.out.println(M + " " + lambda + " " + WR[j]);
          break;
  } } } }

     // Method gauss: pts & wts for Gauss quadrature, uniform [a, b]
  private static void
       gauss(int npts, double a, double b, double[] x, double[] w) {
    int m = (npts + 1)/2;
    double t, t1, pp = 0, p1, p2, p3, eps = 3.e-10;
                                          // eps = accuracy TO CHANGE
    for ( int i=1; i <= m; i++ ) {
      t = Math.cos(Math.PI*(i-0.25)/(npts + 0.5)); t1 = 1;
      while((Math.abs(t-t1))>=eps) {
        p1 = 1. ; p2 = 0. ;
        for ( int j=1; j <= npts; j++ )
          {p3 = p2; p2 = p1; p1 = ((2*j-1)*t*p2-(j-1)*p3)/j;}
        pp = npts*(t*p1-p2)/(t*t-1);
        t1 = t; t = t1 - p1/pp;
      }
      x[i-1] = -t; x[npts-i] = t;
     w[i-1]  = 2./((1-t*t)*pp*pp); w[npts-i] = w[i-1];
    }
    for ( int i=0; i < npts ; i++ ) {
      x[i] = x[i]*(b-a)/2. + (b + a)/2. ;
      w[i] = w[i]*(b-a)/2. ;
} } }
```