Physics 562 - Computational Physics

Final

Josh Fernandes
Department of Physics & Astronomy
California State University Long Beach

May 15, 2014

Abstract

This paper examines three ways of solving for eigenfunctions. The first problem uses Runge-Kutta to solve for the wave functions of a one-dimensional problem with a coulomb potential. The second problem uses matrices to solve for the same problem. The third problem solves a scattering problem.

1 Problem 1

The first system is a quantum one-dimensional problem with $\hbar = 1$ and $mass = 1$. The potential is of the form

$$V(x) = \frac{1}{\sqrt{1+x^2}}. \tag{1}$$

The eigenvalues are calculated by assuming the value of the endpoints and using Runge-Kutta to solve for the wave function. Chebyshev is then used to find the energies where either the wave function or the derivative of the wave function vanish at zero. To find orthogonality, use the equation

$$\int_{-\infty}^{+\infty} \psi_n^* \psi_m \, dx = \delta_{nm}. \tag{2}$$

If $n = m$ integrating over the wave functions will give a value of 1, otherwise integrating will give a value of 0.

1.1 The Fortran95 code

Listing 1: program `ho1d.f95`

```
1
2
3  module setup
4
5      use NumType
6      implicit none
7      integer, parameter  :: n_eq = 3
8      real(dp), parameter :: hbar = 1._dp,hbar2 = hbar**2,&
9                             mass = 1._dp, omega = 1._dp, &
10                            x0 = sqrt(hbar/(mass*omega))
11     real(dp) :: energy, xmax, dx, eps, total,val1,val2
12     real(dp), allocatable, dimension(:,:) :: wf, w1
13     integer :: imax
14
15 end module setup
16
17 program ho1d
18
19     use setup
20     use chebyshev
21     implicit none
22     real(dp) :: eminx, emaxx, emin, emax, de, e0, de0
23     real(dp), external :: psi0
24     integer :: nch, iz, i, maxf, izz,m,n,j,k
25
26     eminx = -1._dp
27     emaxx = 0._dp
28
29     xmax = 40.0_dp
30     dx = 0.01_dp
31     eps = 0.000001_dp
32     de0 = 0.01_dp
33     maxf = 10
34     imax = abs(xmax/dx)
35     allocate(wf(0:imax,2))
36     allocate(w1(0:2*imax,10))
37     de = 0.01_dp
```

```
38      izz = 0
39      nch = 5
40      emin = eminx
41      emax = emin+de
42
43      print *, '================Eigenvalues==============='
44
45      do while (emin < emaxx)
46
47           call chebyex(psi0, nch, cheb, emin, emax)
48           call chebyzero(nch,cheb,emin,emax,z0,iz0)
49
50           do iz=1,iz0
51                e0=z0(iz)
52                call root_polish(psi0,e0,de0,eps,maxf)
53                izz = izz + 1
54                print *, izz, e0
55                call wavef(e0,izz)
56           end do
57
58           emin = emax
59           emax = emin + de
60
61      end do
62
63      print *, '============Orthogonality Test============'
64
65      total = 0
66      val1=0
67      val2=0
68      j=1
69      k=1
70
71      do j = 1,10
72      do k = 1,10
73      do i = 1,2*imax
74           m=j
75           n=k
76           val1=w1(i,m)*w1(i,n)
77           val2=w1(i+1,m)*w1(i+1,n)
```

```
78          !Rombint/trapazoidal method of integration
79          total = total + (0.01*(val1+1/2._dp*(val2-val1)))
80      end do
81
82      print '(a,1f5.1,a5,i2,a5,i2)','total=', &
83           total,'i=',j,'j=',k
84
85      total = 0
86
87      end do
88      end do
89
90
91
92  end program ho1d
93
94  function psi0(eee) result(psi)
95
96      use setup
97      implicit none
98      real(dp), intent(in) :: eee
99      real(dp) :: x, psi
100     real(dp), dimension(n_eq) :: y
101
102     energy = eee
103     x = xmax
104     y(1) = 0.00001
105     y(2) = -0.00001
106     y(3) = 0._dp
107     do while ( x > 0._dp )
108         call rk4step(x,y,-dx)
109     end do
110     psi = y(1)*y(2)
111
112 end function psi0
113
114 subroutine wavef(eee,iz)
115
116     use setup
117     implicit none
```

```
118       real(dp), intent(in) :: eee
119       real(dp) :: x, parity
120       real(dp), dimension(n_eq) :: y
121       integer :: iz, i, imin
122       energy = eee
123       x = xmax
124       y(1) = 0.00001
125       y(2) = -0.00001
126       y(3) = 0._dp
127       do while ( x > 0._dp)
128           call rk4step(x,y,-dx)
129       end do
130
131       x = xmax
132       i = imax + 1
133       y(1) = 0.00001
134       y(2) = -0.00001
135       y(3) = 0._dp
136       do while ( x > 0._dp)
137           i = i-1
138           wf(i,1) = x
139           wf(i,2) = y(1)
140           call rk4step(x,y,-dx)
141       end do
142
143       !establish parity so the negative side of
144       !graph can be plotted
145       imin=i
146       if( abs(y(1)) > abs(y(2)) ) then
147           parity = 1
148       else
149           parity = -1
150       end if
151
152       ! make a master list of wavefunctions called w1.
153       !Used for integration
154       w1(0:imax,iz) = parity*wf(0:imax,2)/sqrt(2*y(3))
155       w1(imax:2*imax,iz) = wf(0:imax,2)/sqrt(2*y(3))
156
157       !normalize the wavefunction
```

```
158     wf(0:imax,2) = wf(0:imax,2)/sqrt(2*y(3))
159
160
161     !make the graphs for positive and negative sides
162     do i = imax, imin, -1
163          write(unit=20+iz,fmt='(2f15.5)') wf(i,1), &
164               wf(i,2)
165     end do
166
167     do i = imin, imax
168          write(unit=20+iz,fmt='(2f15.5)') -wf(i,1), &
169               parity*wf(i,2)
170     end do
171
172
173 end subroutine wavef
```

2 Problem 2

The next problem solves the same system as problem 1, but to solve it in the basis of the simple harmonic oscillator.

2.1 The Fortran95 code

Listing 2: `int.f95`

3 Problem 3

The thirds system solves for a quantum scattering system. The wave function vanishes at zero and as $r \to \infty$ the wave function becomes

$$\psi(r) \sim sin(kr - l\frac{\pi}{2} + \delta_l), \tag{3}$$

where

$$k = \sqrt{\frac{2mE}{\hbar^2}}. \tag{4}$$

The $l = 0$ phase shift are calculated for $E = 1, 5, 10, \& 20$. The bound state energies for $l = 0$ are also computed. This problem required two different codes. One that is run with `ho1d` to calculate the phase shift and one that is run with `ho2d` to calculate the bound state energies.

3.1 The Fortran95 code

Listing 3: `ho1d10.f95`

```
1
2
3  module setup
4
5      use NumType
6      implicit none
7      integer, parameter :: n_eq = 3
8      real(dp), parameter :: hbar2=1._dp, &
9              mass=1.0_dp, xm=mass*mass/(mass+mass)
10     integer :: l
11
12     real(dp) :: energy, xmax, dx, eps, xmid
13     real(dp), allocatable, dimension(:,:) :: wf
14     integer :: imax
15
16 end module setup
17
18 program ho1d
19
20     use setup
21     use chebyshev
22     implicit none
23     real(dp) :: dmin,dmax,de,delta0,psi, energy_array(4)
24     real(dp), external :: psi0
25     integer :: nch, iz, i, maxf
26
27     l=0
28     energy_array(1)=1._dp
29     energy_array(2)=5._dp
30     energy_array(3)=10._dp
31     energy_array(4)=20._dp
```

```
32
33      xmax=10._dp
34      dx=0.001_dp
35      eps=0.0000001_dp
36      maxf=20
37      dmin=0._dp
38      dmax=pi
39
40      imax=nint(xmax/dx)+1
41      allocate(wf(0:imax,2))
42
43      nch=6
44
45      do i = 1,4
46      energy = energy_array(i)
47      print *,'for Energy=', nint(energy)
48      call chebyex(psi0,nch,cheb,dmin,dmax)
49      call chebyzero(nch,cheb,dmin,dmax,z0,iz0)
50
51      de=0.01_dp
52
53      do iz=1,iz0
54          delta0=z0(iz)
55          call root_polish(psi0,delta0,de,eps,maxf)
56          psi=psi0(delta0)
57          print *, ' Delta=', delta0
58          call wavefunction(delta0,i)
59      end do
60
61      end do
62
63  end program ho1d
64
65  function psi0(delta) result(psi)
66
67      use setup
68      implicit none
69      real(dp), intent(in) :: delta
70      real(dp) :: x, psi, k
71      real(dp), dimension(n_eq) :: y
```

```
72
73      k = sqrt(2*xm/hbar2*energy)
74
75      x=xmax
76      y(1) = sin(k*x - l*pi/2 +delta)
77      y(2) = k*cos(k*x -l*pi/2 +delta)
78      do while ( x>xmid)
79          call rk4step(x,y,-dx)
80      end do
81      psi=y(2)
82
83  end function psi0
84
85  subroutine wavefunction(delta,index)
86
87      use setup
88      implicit none
89      real(dp), intent(in) :: delta
90      integer, intent(in) :: index
91      real(dp) :: x, k, y12,y11
92      real(dp), dimension(n_eq) :: y
93      integer :: n, i
94
95      k=sqrt(2*xm/hbar2*energy)
96
97      x=xmax
98      y(1)=sin(k*x-l*pi/2+delta)
99      y(2)=k*cos(k*x-l*pi/2+delta)
100     do while (x>xmid)
101         n=nint(x/dx)
102         wf(n,1)=x
103         wf(n,2)=y(1)
104         call rk4step(x,y,-dx)
105     end do
106     y12=y(1)
107
108     wf(0,1)=0._dp
109     wf(0,2)=0._dp
110     x=dx
111     y(1)=x**(l+1)
```

```
112     y(2)=(l+1)*x**l
113     do while (x<=xmid)
114         n=nint(x/dx)
115         wf(n,1)=x
116         wf(n,2)=y(1)
117         call rk4step(x,y,dx)
118     end do
119     y11=y(1)
120
121     wf(0:n,2)=y12/y11*wf(0:n,2)
122     n=nint(xmax/dx)
123     do i = 0,n
124         write(10+index,*) wf(i,1),wf(i,2)
125     end do
126
127 end subroutine wavefunction
```

Listing 4: ho1d10_energy.f95

```
1
2
3  module setup
4
5      use NumType
6      implicit none
7      integer, parameter :: n_eq = 3
8      real(dp), parameter :: hbar2=1._dp, &
9              mass=1.0_dp, xm=mass*mass/(mass+mass)
10     integer :: l
11
12     real(dp) :: energy, xmax, dx, eps, xmid
13     real(dp), allocatable, dimension(:,:) :: wf
14     integer :: imax
15
16 end module setup
17
18 program bound
19
20     use setup
21     use chebyshev
22     implicit none
```

```
23      real(dp) :: eminx,emin,emaxx,emax,deltae, &
24          de,e0,psi
25      real(dp), external :: psi0
26      integer :: nch, izz, i, maxf, n, nstep
27
28      l=0
29
30      xmax=5.0_dp
31      dx=0.001_dp
32      eps=0.0001_dp
33      maxf=20
34      eminx=-100._dp
35      emaxx=0._dp
36      de=0.2_dp
37      nstep=5
38      nch=5
39
40      imax=nint(xmax/dx)+1
41      allocate(wf(0:imax,2))
42
43      e0=eminx
44      do
45          psi=psi0(e0)
46          if(psi /= xmax .or. e0 > emaxx) exit
47          e0=e0+de
48      end do
49
50      eminx=e0
51  !     print *, eminx
52      deltae=(emaxx-eminx)/nstep
53  !     print *, deltae
54
55      izz = 0
56
57      print *, 'The bound state energy are '
58      do n=1,nstep
59
60          emin = eminx+(n-1)*deltae
61          emax = eminx+n*deltae
62  !         print *, emin, emax
```

```
63          call chebyex(psi0,nch,cheb,emin,emax)
64          call chebyzero(nch,cheb,emin,emax,z0,iz0)
65  !         print *, z0(1:iz0)
66
67          de=0.1_dp
68          do i=1,iz0
69              e0=z0(i)
70              call root_polish(psi0,e0,de,eps,maxf)
71              psi=psi0(e0)
72              izz=izz+1
73              print *, izz,'E=',e0
74              call wavef(e0,izz)
75          end do
76
77      end do
78
79  end program bound
80
81  function psi0(eee) result(psi)
82
83      use setup
84      implicit none
85      real(dp), intent(in) :: eee
86      real(dp) :: x, psi, k
87      real(dp), dimension(n_eq) :: y
88
89      energy = eee
90      k = sqrt(2*xm/hbar2*(-energy))
91
92      x=dx
93      y(1)=x**(l+1)
94      y(2)=(l+1)*x**l
95      y(3)=0._dp
96      do while (x <= xmax .and. y(2) > 0._dp)
97          call rk4step(x,y,dx)
98      end do
99
100     xmid=x
101     if ( xmid >= xmax ) then
102         psi=xmax
```

```
103             return
104         end if
105
106         x=xmax
107         y(1) = exp(-k*x)
108         y(2) = -k*y(1)
109         y(3) = 0._dp
110         do while (x > xmid)
111             call rk4step(x,y,-dx)
112         end do
113         psi=y(2)
114
115 !       print *, e,psi
116
117 end function psi0
118
119 subroutine wavef(eee,iz)
120
121         use setup
122         implicit none
123         real(dp), intent(in) :: eee
124         real(dp) :: x, psi,k,y12,y32,y11,y31,yy
125         real(dp), dimension(n_eq) :: y
126         integer :: iz, n, i
127
128         energy = eee
129         k=sqrt(2*xm/hbar2*(-energy))
130
131         x=xmax
132         y(1)=exp(-k*x)
133         y(2)=-k*y(1)
134         y(3) = 0._dp
135         do while (x>xmid)
136             n=nint(x/dx)
137             wf(n,1)=x
138             wf(n,2)=y(1)
139             call rk4step(x,y,-dx)
140         end do
141         y12=y(1)
142         y32=-y(3)
```

```
143
144       wf(0,1)=0._dp
145       wf(0,2)=0._dp
146       x=dx
147       y(1)=x**(l+1)
148       y(2)=(l+1)*x**l
149       y(3)=0._dp
150       do while ( x <= xmid )
151            n=nint(x/dx)
152            wf(n,1)=x
153            wf(n,2)=y(1)
154            call rk4step(x,y,dx)
155       end do
156       y11=y(1)
157       y31=y(3)
158
159       wf(0:n,2)=y12/y11*wf(0:n,2)
160       y31=(y12/y11)**2* y31
161       yy=y31+y32
162       n=nint(xmax/dx)
163       wf(0:n,2)=wf(0:n,2)/sqrt(yy)
164       do i = 0,n
165            write(30*(l+1)+iz,*) wf(i,1),-wf(i,2)
166       end do
167
168  end subroutine wavef
```

4 Results

4.1 Problem 1

The results from problem 1 give the following energies,

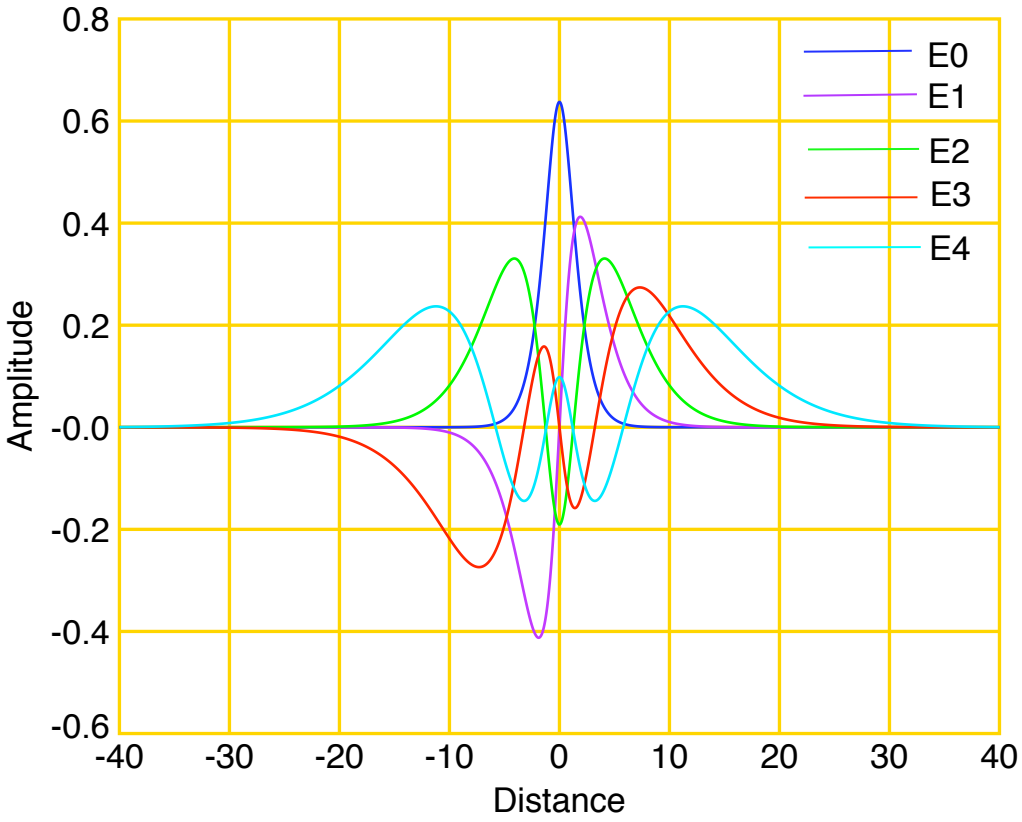

Figure 1: The first five wave functions for problem 1

$$E_0 = -0.669777$$
$$E_1 = -0.274891$$
$$E_2 = -0.151454$$
$$E_3 = -0.092679$$
$$E_4 = -0.063526$$
$$E_5 = -0.045477$$
$$E_6 = -0.034374$$
$$E_7 = -0.025277$$
$$E_8 = -0.016334$$
$$E_9 = -0.005216$$

Notice that as you go up in energy values, the spacing between energy levels gets smaller. The first five wave functions plotted in Fig 1. The eigenstates are also orthogonal as the relationship in Eq. 2 holds true.

4.2 Problem 3

For problem 3, the phase shifts for their respective energies are

$$E = 1, \delta = 2.3811$$
$$E = 5, \delta = 1.840$$
$$E = 10, \delta = 1.232$$
$$E = 20, \delta = 0.719$$

A completely free particle has a phase shift $\delta_l = 0$. The results show that as the energy increases, δ_l approaches zero. The reason for this is that as a particle has a higher energy, the more easily it can escape from a potential well. The scattering wave functions are plotted in Fig. 2. The bound states for energies for $l = 0$ are

$$E_0 = -26.7309$$
$$E_1 = -13.6180$$
$$E_2 = -3.29810$$

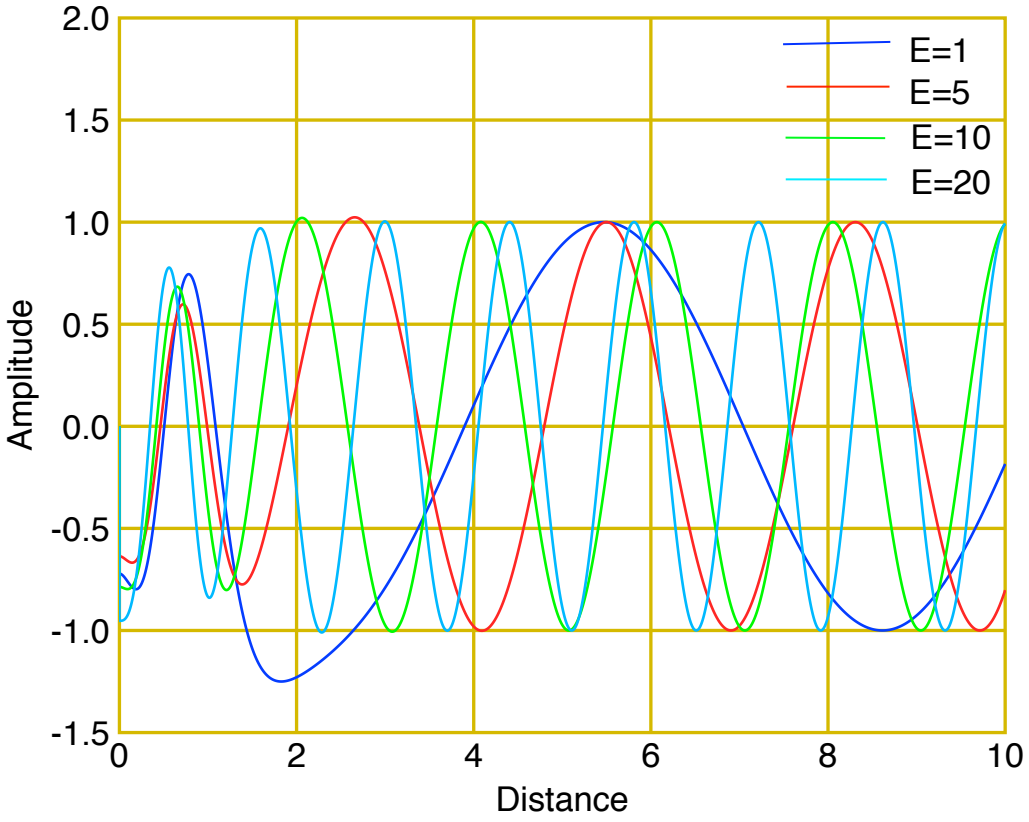

Figure 2: The scattering functions for problem 3

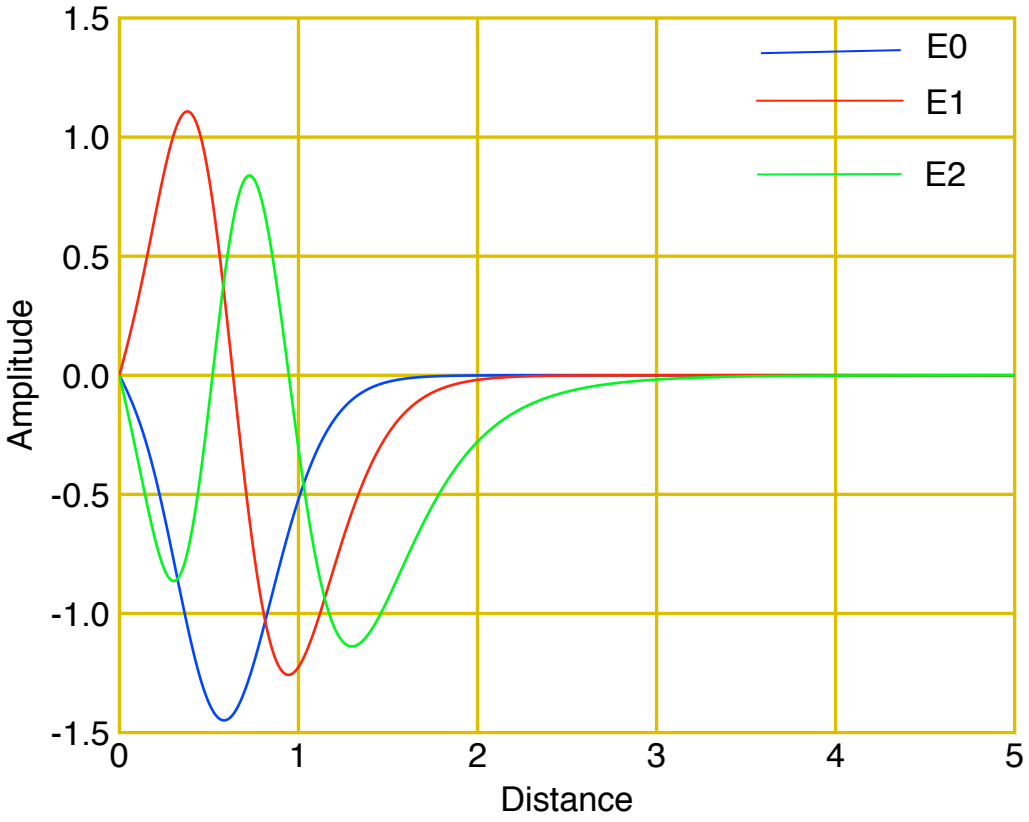

Figure 3: The bound functions for problem 3

References

[1] M. Metcalf, J. Reid and M. Cohen, *Fortran 95/2003 explained*. Oxford University Press, 2004.