Práctica Computacional: La Ecuación de Schrödinger Estados ligados [§]

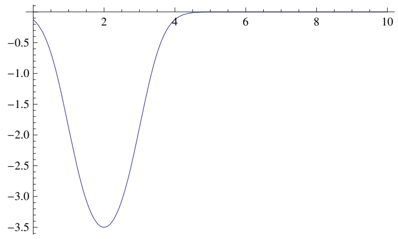
A. Parte computacional

1. Utilizar el *notebook* LIGADOS.NB para obtener algunos estados ligados de una partícula en un potencial de *Woods-Saxon.*

```
(* Potencial de Woods-Saxon *)

In[1]:= V[x_] := - U/(e^((x-a)^2) + 1); U = 7.; a = 2.;
        Plot[V[x], {x, 0.01, 10}, PlotRange → All]
```

Out[2]=

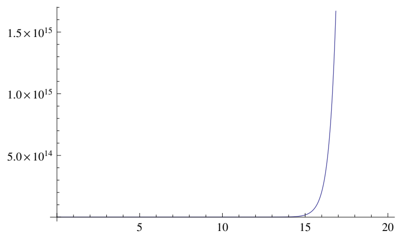



```
(* Busqueda de Estados Ligados *)

In[3]:= Ener = -3;
        solution2 = NDSolve[{- 1./2. * y''[x] + V[x] y[x] == Ener * y[x],
            y'[0.] == 1, y[0.] == 0.}, y, {x, 0, 20}]
        Plot[Evaluate[{y[x]} /. solution2], {x, 0, 20}]

Out[4]= {{y → InterpolatingFunction[{{0., 20.}}, <>]}}
```

Out[5]=

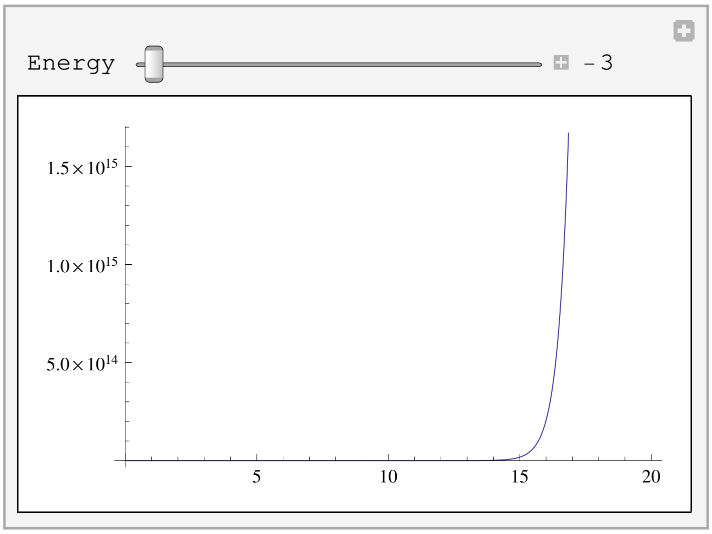


[§] http://www.df.uba.ar/users/dmitnik/computation/ligados/ligados.html

(* Busqueda automatica de estados ligados*)

In[6]:=
```
Manipulate[
 solution2 = NDSolve[{- 1./2. * y''[x] + V[x] y[x] == Ener * y[x] ,
    y'[0.] == 1, y[0.] == 0.}, y, {x, 0, 20}] ;
Plot[Evaluate[{y[x]} /. solution2], {x, 0, 20}] ,
 { {Ener, -3, "Energy"}, -3, -2, -0.05, Appearance → "Labeled"}]
```

Out[6]=



Ajuste fino

In[7]:=
```
Manipulate[
 solution2 = NDSolve[{- 1./2. * y''[x] + V[x] y[x] == Ener * y[x] ,
    y'[0.] == 1, y[0.] == 0.}, y, {x, 0, 20}] ;
Plot[Evaluate[{y[x]} /. solution2], {x, 0, 10}, PlotRange → {-5, 5}] ,
 { {Ener, -2.6, "Energy"}, -3, -2, -0.00005, Appearance → "Labeled"}]
```

B. Método de Shooting (Predictor)

La base de este método consiste en partir de cierta condición inicial, y propagar la solución hasta un radio determinado. Se discretizan la función y sus derivadas, escribiendo la derivada primera de una función Y_i (o sea, la función Y evaluada en el punto i):

$$Y_i^{'} = \frac{Y_{i+1} - Y_i}{\Delta x}$$

Donde Δx es el ancho del intervalo ($\Delta x = x_{i+1} - x_i$). Del mismo modo, derivando una vez mas, la derivada segunda se escribe:

$$Y_i^{''} = \frac{Y_{i+1} + Y_{i-1} - 2Y_i}{(\Delta x)^2}$$

Implementaremos este método para resolver el problema de un pozo finito, de ancho D y altura V_0, usando el programa SCHRO.FOR:

```
      program schrodinger

c...... Solves the time-independent Schrodinger equation
c...... for a potential box. Uses simple finite differences

c       by Dario Mitnik

      implicit real*8(a-h,o-z)


c....... initial values
      print*,' give the number of grid points'
      read*,npts
c.......   Potential Box
      print*,' give the size D of the potential box (-D/2 to D/2)'
      read*,d
      print*,' give the depth V0 of the potential well'
      read*,V0
c......... Initial Points
      print*,' give the initial value of the wavefunction P(0) '
      read*,p00
      print*,' give the initial value of the derivative at 0 Y(0) '
      read*,y00


100     open(unit=10,file='wave.dat',status='unknown')
      print*,' guess initial energy value'
      read*,Ener
      p0 = p00
      y0 = y00

c...... Other values
      dx = D/(2*npts)
c...... initial point P(1)
      p1 = p0 + dx*y0
      write(10,55) dx-dx,p0
      write(10,55) dx,p1
55       format(2(1x,1pg14.4))

c...... Solving the Schrodinger Equation
      do 200 i=2,npts*15
```

```
            V = V0
            if (i*dx.le.D/2) V = 0.0
            p = (2.0 + dx*dx*(V-Ener))*p1 - p0
            write(10,55) dx*i,p
            p0 = p1
            p1 = p
200     continue

       close(unit=10,status='keep')

       print*,' Continue?  No=0'
       read*,icont
       if (icont.ne.0) go to 100

       stop
       end
```

Tal como se puede ver en el listado del programa, este requiere los datos del pozo (ancho D y altura V_0), y los datos iniciales de la función de onda. Luego se deben ingresar distintos valores de la energía (es lo que debemos hallar!), e ir variando éstos hasta que la solución converja a un resultado adecuado. Inicialmente el programa utiliza los valores de la función en los dos primeros puntos (p_0 y p_1), y calcula el valor para el tercer punto (p). Luego, utilizando los puntos 2 y 3, calcula el valor para el punto 4, y así sucesivamente. En el gráfico siguiente, podemos ver un ejemplo del uso de este programa, y como se ven las soluciones, adivinando diferentes valores de la energía:

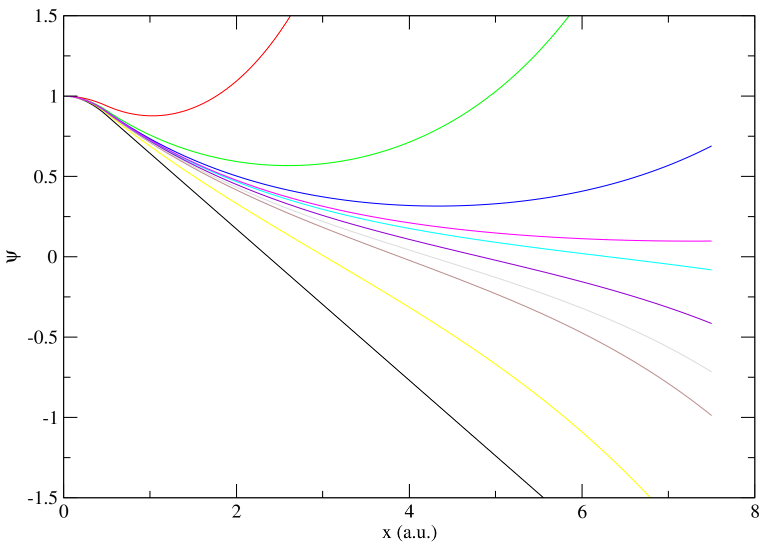


FIG. 1. Búsqueda de los estados ligados con el método de shooting.

1. Ejercicios

1. Compilar y ejecutar el programa, revisar si las soluciones dadas son correctas, y si hay algn error en el programa, modificarlo hasta que d resultados adecuados!.
2. Graficar los resultados
3. ¿Cuńtos estados ligados hay?
4. Aumentar progresivamente la profundidad del pozo de potencial, hasta encontrar más estados ligados:
   (a) ¿Qué ocurre con las energías?
   (b) ¿Qué ocurre con el número de nodos?
5. Aumentar progresivamente el ancho del pozo de potencial:
   (a) ¿Cómo cambian las funciones de onda?
   (b) ¿Qué ocurre con las energías?
   (c) ¿Se puede determinar la profundidad del pozo conociendo sólo la energía del estado básico? Demostrarlo
6. Analizar la paridad de las funciones obtenidas
7. Modificar el programa de manera que se puedan ingresar los números de nodos
8. Modificar el programa y solucionar la ecuación de Schrödinger para otro potencial
9. Comparar (graficando) los resultados con los resultados analíticos
10. Implementar un algoritmo tal que el programa encuentre por sí mismo las energías correctas de los estados ligados (Solución: tratar de **no ver** el programa NUMEROVBOUND.FOR).

C. Método Predictor–Corrector

En éste método no sólo se propaga la solución en "sendito saliente", sino que luego se calcula otra solución "entrante" y se las compara. El programa NUMEROVBOUND.FOR ilustra el funcionamiento del método:

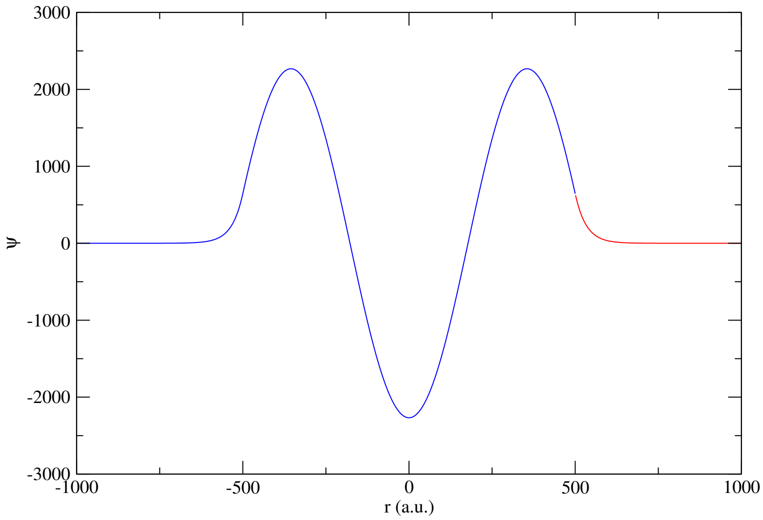


FIG. 2. Búsqueda de los estados ligados con el método de shooting.

1. Ejercicios

1. Escribir un programa para calcular las funciones de onda del átomo de hidrógeno. (Solución: tratar de **no ver** el programa HYDRONUMER.FOR)

2. Escribir un programa para calcular las funciones de onda del átomo de hidrógeno, pero en este caso, **las energías no son datos**:

```
c-------finding the energy using the Ridley Method
c.......E.C. Ridley, Proc. Cambridge Phil. Soc. 51, 702 (1955).

c Delta E = ( logderiv(inwd) - logderiv(outw) )
c   -----------------------------------
c   ( Intgrl(inwd^2) + Intgrl(outw^2) )
c     --------------   --------------
c         inwd^2           outw^2

c logderiv(outw) = F'/F at r=rmatch
c logderiv(inwd) = G'/G at r=rmatch
c Intgrl(outw^2) = Int_{0}^{rmatch}{F^2}
c Intgrl(inwd^2) = Int_{rmatch}^{rmax}{G^2}
c F is the outward solution and G is the inward solution
```