VQMC (Variational Quantum MonteCarlo): Àtom d'hidrogen

Josep Planelles

February 12, 2020

L'hamiltonià de l'àtom d'hidrogen

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2mr^2}\left[\left(\frac{\partial}{\partial r}\left(r^2\,\frac{\partial}{\partial r}\right)\right) - \frac{\widehat{L^2(\theta,\phi)}}{\hbar^2}\right] - \frac{1}{r} \tag{1}$$

dóna lloc a l'equació d'autovalors per a estats esfèrics ($\ell = 0$):

$$-\frac{\hbar^2}{2mr^2}\left[\frac{\partial}{\partial r}\left(r^2\,\frac{\partial}{\partial r}\right)\right] R(r) - \frac{1}{r}\,R(r) = E\,R(r) \tag{2}$$

Per tant, per als estats amb $\ell = 0$, l'Hamiltonià és simplement (en a.u., $\hbar = m = 1$)

$$\hat{\mathcal{H}}_{\ell=0} = -\frac{1}{2r^2}\left[\frac{\partial}{\partial r}\left(r^2\,\frac{\partial}{\partial r}\right)\right] - \frac{1}{r} \tag{3}$$

La solució no normalitzada de l'estat fonamental $1s$ és $R_{1,0}(r) = e^{-r}$.

La probabilitat radial no normalitzada $P(r) = r^2\,e^{-2r}\,dr$.

Una funció variacional no normalitzada apropiada podria ser $R_\alpha(r) = e^{-\alpha r}$.

La probabilitat radial no normalitzada $P_\alpha(r) = r^2\,e^{-2\alpha r}\,dr$.

L'energia local associada $E^{(\alpha)}_{Loc}(r) = \frac{1}{R_\alpha(r)}\,\hat{\mathcal{H}}_{\ell=0}\,R_\alpha(r) = -\frac{1}{r} - \frac{\alpha}{2}\,\left(\alpha - \frac{2}{r}\right)$.

El principi variacional diu que l'energia exacta E_0 de l'estat fonamental és una cota inferior del valor expectació de l'hamiltonià $\hat{\mathcal{H}}_{\ell=0}$ amb la funció variacional (normalitzada):

$$\begin{aligned} E_0 \le \langle E\rangle_\alpha &= \frac{\int_0^\infty R_\alpha(r)\hat{\mathcal{H}}_{\ell=0}R_\alpha(r)r^2dr}{\int_0^\infty R_\alpha(r)R_\alpha(r)r^2dr} = \frac{\int_0^\infty R_\alpha(r)^2\left[\frac{1}{R_\alpha(r)}\hat{\mathcal{H}}_{\ell=0}R_\alpha(r)\right]r^2dr}{\int_0^\infty R_\alpha(r)^2r^2dr} = \frac{\int_0^\infty R_\alpha(r)^2E^{(\alpha)}_{Loc}(r)r^2dr}{\int_0^\infty R_\alpha(r)^2r^2dr} \\ &= \frac{\int_0^\infty P_\alpha(r)E^{(\alpha)}_{Loc}(r)dr}{\int_0^\infty P_\alpha(r)dr} \end{aligned} \tag{4}$$

amb $P_\alpha(r) = r^2\,e^{-2\alpha r}$ la probabilitat radial no normalitzada.

Per tal d'avaluar estadísticament la integral, generem una posició aleatòria dintre d'un interval prefixat Δ_0 (per exemple 4 a.u.). L'anomenem r_1, calculem la seua probabilitat no normalitzada $P_\alpha(r_1)$ i fiquem en una llista la seua energia local $E^{(1)}_{Loc}$. Tot seguit generem una altra posició aleatòria r_2 i calculem la seua probabilitat

no normalitzada $P_\alpha(r_2)$. Tot seguit generem un número aleatori $0 \leq w \leq 1$ i comparem la ratio $\frac{P_\alpha(r_2)}{P_\alpha(r_1)}$ amb w. Si $\frac{P_\alpha(r_2)}{P_\alpha(r_1)} \geq w$ fiquem l'energia local $E^{(2)}_{Loc}$ en la llista d'energies. En cas contrari se rebutja el punt i se genera un altre punt amb el qual fem la comparació. El procés se continua un nombre prefixat de vegades (e.g. $N = 100.000$) i avaluem $\langle E \rangle_\alpha$ mitjançant l'equació:

$$\langle E \rangle_\alpha = \frac{1}{N_a} \sum_i E^{(i)}_{Loc} \tag{5}$$

on $N_a < N$ és el nombre de valors guardats en la llista.
Si paral·lelament, en una segon llista guardem valors $(E^{(i)}_{Loc})^2$, podem estimat la variància de la mostra:

$$v = \langle (E^{(i)}_{Loc})^2 \rangle - \langle E^{(i)}_{Loc} \rangle^2 \tag{6}$$

Finalment, per evitar arbitrarietat en l'interval Δ_0 prefixat procedim a *termalitzar* la mostra. Cosa que vol dir que en una etapa prèvia al càlcul i magatzematge d'energies i variàncies, assumim un interval i procedim al proces de de comparació de probabilitats en una sèrie $M \approx 0.2N$ de punts aleatoris. Únicament contem el nombre m_a de vegades que acceptaríem el nou punt. Aleshores renormalitzem l'interval prefixat amb la condició que siga tal que acceptaríem tants punts m_a com punts m_r rebutjaríem, és a dir $m_a = m_r = M/2$. Si $m_a \neq m_r$ procedim a canviar Δ_0:

$$\Delta = \Delta_0 \frac{m_a}{M/2} \tag{7}$$

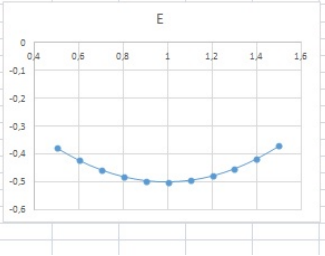
És a dir, si hi ha més encerts que la meitat $M/2$ de punts fem l'interval més llarg (assolint regions de menor probabilitat que causarien més rebutjos). En cas contrari faríem l'interval més curt.

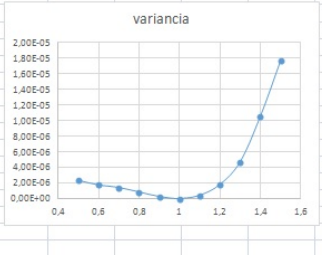
Podem obtenir major precisió repetint k vegades el procés i calculant mitjanes (en l'argot de VQMC diríem que tenim k *walkers*).

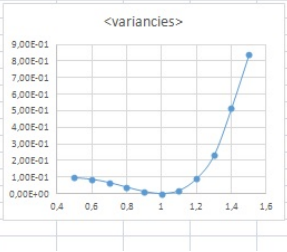
La figura mostra alguns resultats que hem obtingut amb un programeta fortran:

r0	m	walkers	sec	encerts
4	1,E+05	300	1	0,5

α	E	variancia	<variancies>
0,5	-0,376805	2,356E-06	9,55E-02
0,6	-0,421608	1,756E-06	8,56E-02
0,7	-0,456232	1,438E-06	6,58E-02
0,8	-0,480963	8,59E-07	3,82E-02
0,9	-0,495624	2,72E-07	1,24E-02
1	-0,5	0	0,00E+00
1,1	-0,494342	4,17E-07	1,81E-02
1,2	-0,478438	1,794E-06	8,99E-02
1,3	-0,452488	4,678E-06	2,30E-01
1,4	-0,416187	1,052E-05	5,15E-01
1,5	-0,370392	1,767E-05	8,37E-01







1 Apèndix: el programa f90

El programa f90 bàsicament consta de tres *functions* que són invocades reiteradament i que fan el càlcul de l'energia, la probabilitat i el número random:

```
double precision function ENE(x)
Implicit double precision(a-h,o-z)
common/dades1/alpha,r0,pas


ENE =-1.d0/x-(alpha/2.d0)*(alpha-2.d0/x)

return
end
```

```
double precision function prob(x)
Implicit double precision(a-h,o-z)
common/dades1/alpha,r0,pas

prob =  exp(-2*alpha*x)*x**2

return
end
```

```
double precision function ran2(idum)
Implicit double precision(a-h,o-z)
call random_number ( x_scalar )
ran2=x_scalar
return
end
```

El càlcul es fa bàsicament en la routina *walk*, la qual en primer lloc procedeix a fer la temalització

```
      subroutine walk_scan
      Implicit real*8(a-h,o-z)
      double precision:: alpha,r0,pas
      integer:: cont,cont2,walkers, m, mi
      common/dades1/alpha,r0,pas
      common/dades2/walkers, m, mi, sec,encerts
      common/results/ene1,ene2
      double precision, allocatable :: llista(:),llista2(:)

      allocate ( llista(m),llista2(m))

      x=(r0/2.d0)*ran2(idum)
      p0= prob(x)

! termalitzation

    cont=0
    do i=1,mi

      pastemx=pas*ran2(idum)
!     xb=x+pastemx
      xb=pastemx
      p1=prob(xb)
      w=ran2(idum);

      IF(p1/p0>sec*w) THEN
        x=xb
        p0=p1
        cont=cont+1
      ENDIF

    enddo
```

i continua amb el càlcul d'energies i variances:

```
!     write(*,*) "initial step ",pas
        pas=pas*cont/(encerts*mi);
!     write(*,*) " fitted step ",pas

    cont=0
    cont2=0

    do i=1,m
      pastemx=pas*ran2(idum)
      xb=pastemx
      p1=prob(xb)
      w=ran2(idum);

      IF(p1/p0>sec*w) THEN
        cont=cont+1
        aux=ene(xb)
        llista(cont)=aux
        llista2(cont)=aux**2
        x=xb
        p0=p1
      ELSE
        cont2=cont2+1
      ENDIF

    enddo

        ene1=sum(llista(1:cont))/cont
        ene2=sum(llista2(1:cont))/cont

    deallocate (llista, llista2)
  return
  end
```

Els bucles, lectures d'*input* i escriptura d'*outputs* se fa en el programa principal:

```
      Implicit real*8(a-h,o-z)
      integer:: walkers, m, mi
      common/dades1/alpha,r0,pas
      common/dades2/walkers, m, mi, sec,encerts
      common/results/ene1,ene2
      double precision, allocatable :: llistene(:), llistvar(:), llistene2(:)

      elv = 27.211385d0
      ban = 0.529177249d0
      pi=dacos(-1.d0)

      open (unit=10, file='input.data', status='old')
      read(10,*)
      read(10,*) r0,m,walkers,sec,encerts
      read(10,*)
      read(10,*)  alphai,alphaf,alphap
      close(10)

      open(unit=40,file='out.data')
      write(40,*) "r0,m,walkers,sec,encerts"
      write(40,*)  r0,",",m," ",walkers," ",sec," ",encerts
      write(40,*) " alphai,alphaf,alphap"
      write(40,*) alphai,alphaf,alphap
      write(40,*)  "{alpha,energia,variancia per al calcul d'errors,mitjana de variancies en els walks}"


      allocate (llistene(walkers), llistvar(walkers), llistene2(walkers))
      mi=int(0.2 * m)
      pas = r0
      t1=secnds(time)


   do alpha=alphai,alphaf,alphap

      t0=secnds(time)
      do kk=1,walkers

         call walk_scan

         llistene(kk)=ene1
         llistene2(kk)=ene1**2
         llistvar(kk)=ene2-ene1**2
      enddo

      energia=sum(llistene(1:walkers))/walkers
      energia2=sum(llistene2(1:walkers))/walkers
      varerror=energia2-energia**2
      variancia=sum(llistvar(1:walkers))/walkers
      write(40,*)  "{",alpha,",",energia,",",varerror,",",variancia,"},"
      write(40,*)
   enddo

   deallocate (llistene, llistvar, llistene2)

   temps=secnds(time)-t1
   write(40,*) "temps de calcul",temps
   close(40)
   write(*,*) "temps de calcul",temps
   stop
   end
```