Exercise Lecture X

Variational Monte Carlo for lattice models

1. **The one-dimensional quantum antiferromagnetic Heisenberg model**

We want to compute the variational energy of the Heisenberg $S = 1/2$ Heisenberg model on a chain with L sites (including a spin anisotropy):

$$\mathcal{H} = J_z \sum_{i=1}^{L} S_i^z S_{i+1}^z + \frac{1}{2} J_{xy} \sum_{i=1}^{L} \left(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+ \right), \tag{1}$$

where $J_z \geq 0$ and $J_{xy} = 1$ (fixing the energy units) and periodic boundary conditions are considered, i.e., $S_{L+1}^\mu \equiv S_1^\mu$.

An accurate variational wave function to describe the ground-state properties can be defined in the basis of configurations $\{|\sigma_1, \dots, \sigma_L\rangle\}$ with definite values of spin along the z axis on each site i (e.g., $S_i^z|\sigma_1, \dots, \sigma_L\rangle = s_i|\sigma_1, \dots, \sigma_L\rangle$, where $s_i = \pm 1/2$) and a zero total spin along z (e.g., $S_{\text{tot}}^z = \sum_{i=1}^{L} s_i = 0$). In this basis the wave function is written as

$$\Psi(\{s_i\}) = \text{Sign}(\{s_i\}) \times \exp\left\{ \frac{\alpha}{2} \sum_{i \neq j} v_{i,j} (s_i + 1/2)(s_j + 1/2) \right\}, \tag{2}$$

where

$$\text{Sign}(\{s_i\}) = (-1)^{\sum_{i \in B} (s_i + 1/2)} \tag{3}$$

is the so-called Marshall sign, determined by the number of $s_i = +1/2$ on the sites of the B sublattice (i.e., $i = 2n$) and α is a variational parameter. The pseudo-potential $v_{i,j}$ depends by the chord distance:

$$d_{i,j} = \frac{L}{\pi} \sin\left(\frac{\pi |i - j|}{L} \right) \tag{4}$$

between two sites i and j and is given by

$$v_{i,j} = \ln\left(d_{i,j}^2 \right) \tag{5}$$

For the isotropic case $J_z = J_{xy} = 1$:

- Compute the variational energy $E_0(L)$ for a few cluster sizes (for example $L = 20$, 40, 80, and 160), finding the best value of the variational parameter α (for each size).
- Perform a size scaling of the variational energy per site $\epsilon_0(L) = E_0(L)/L$, obtaining the leading size corrections, i.e., $\epsilon_0(L) = \epsilon_0(\infty) + c/L^\beta + \dots$
- Consider $S_{\text{tot}}^z = 1$ and repeat the calculations to evaluate the energy gap $\Delta E(L) = E_1(L) - E_0(L)$.

- Perform a size scaling of $\Delta E(L)$. Is $\Delta E(\infty)$ finite or vanishing?
- Compute the longitudinal spin-spin correlations for a given size of the cluster and fit them (is it a power-law or an exponential decay with distance?):

$$\mathcal{S}^z(r) = \langle \frac{1}{L} \sum_i S_i^z S_{i+r}^z \rangle$$

- (Optional) Modify the code to compute the in-plane spin-spin correlation functions:

$$\mathcal{S}^{xy}(r) = \langle \frac{1}{2L} \sum_i \left(S_i^x S_{i+r}^x + S_i^y S_{i+r}^y \right) \rangle$$

- (Optional) What about the case with $J_z = 0$ (the so-called XY model)?

```
      program jastrow
      implicit none
      INTEGER(4) nh,sztot
      INTEGER(4) ngen,nscra,nbra,ncorr
      INTEGER(4) i,j,jn,iout,jout,indvic
      INTEGER(4) nacc
      REAL(8) jperp,ener,alpha
      REAL(8) ratio,zeta,rata,rnd

      INTEGER(4), dimension(8) :: iseed
      INTEGER(4), dimension(:,:), allocatable :: ivic
      INTEGER(4), dimension(:), allocatable :: iconf

      REAL(8), dimension(:), allocatable :: tabpip
      REAL(8), dimension(:), allocatable :: szsz
      REAL(8), dimension(:,:), allocatable :: vpot,vjas

      namelist /lattice/ nh
      namelist /parameters/ jperp,sztot
      namelist /wavefunction/ alpha
      namelist /montecarlo/ iseed,ngen,nbra,nscra,ncorr

! reading part
      nh=0
      jperp=1.d0
      sztot=0
      alpha=1.d0

      ngen=0
      nbra=0
      nscra=0
      ncorr=0

      read(5,lattice)
      read(5,parameters)
      read(5,wavefunction)

      if(nh==0) then
       write(6,*) 'nh must be specified'
       stop
      endif

      write(6,*) ' Number of sites                    :',nh
      write(6,*) ' Total Sz                           :',sztot
      write(6,*) ' Anisotropy of the super-exchange   :',jperp
      write(6,*) ' Variational parameter alpha        :',alpha

      read(5,montecarlo)

      if(ngen==0) then
       write(6,*) 'ngen must be specified'
       stop
      endif
      if(nbra==0) then
       write(6,*) 'nbra must be specified'
       stop
      endif
      if(nscra==0) then
       write(6,*) 'nscra must be specified'
       stop
      endif
      write(6,*)
      write(6,*) ' Number of measures                 :',ngen
      write(6,*) ' Number of MC steps between measures:',nbra
      write(6,*) ' Number of measures between upscra  :',nscra
      if(ncorr==0) then
       write(6,*) ' No correlation functions computed'
      else
       write(6,*) ' Correlation functions computed'
      endif

      call random_seed(put=iseed)

      open(unit=11,file='fort.11',form='formatted',status='unknown')
      open(unit=12,file='fort.12',form='unformatted',status='unknown')
      if(ncorr/=0) then
       open(unit=13,file='fort.13',form='unformatted',status='unknown')
      endif

      rewind(11)
      rewind(12)
      if(ncorr/=0) rewind(13)

      ALLOCATE(ivic(nh,2))
      ALLOCATE(vpot(nh,nh))
      ALLOCATE(vjas(nh,nh))
      ALLOCATE(szsz(nh))
      ALLOCATE(tabpip(nh))
      ALLOCATE(iconf(nh))

! table of nearest neighbors
```

```
      call neighbors(nh,ivic)

! pseudo-potential of the Jastrow
      call pseudo(nh,alpha,vpot,vjas)

! random initialization of spins
      call init(nh,sztot,iconf)

! main VMC loop
      nacc=0

      do i=1,ngen

       if(mod(i,nscra)==1) then
        call upscratch(nh,iconf,vjas,tabpip)
       endif

       do j=1,nbra

! nearest-neighbor spin flip
        call random_number(rnd)
        iout=rnd*nh+1
        call random_number(rnd)
        indvic=rnd*2+1
        jout=ivic(iout,indvic)

        call ratiovar(iout,jout,nh,iconf,tabpip,vjas,ratio)

        call random_number(rnd)
        zeta=1.d0-rnd
        if(ratio**2>zeta) then
         nacc=nacc+1
         call upjastrow(nh,iout,jout,iconf,tabpip,vjas)
        endif

       enddo

       call localenergy(nh,iconf,ivic,tabpip,vjas,jperp,ener)

       write(11,*) i,ener/nh
       write(12) i,ener/nh

       if(ncorr/=0) then
        call spinspinz(nh,iconf,szsz)
        write(13) i,(szsz(j),j=1,nh)
       endif
      enddo

      rata=dble(nacc)/(ngen*nbra)
      write(6,*)
      write(6,*) 'accept. rate off diagonal moves =',rata

      close(11)
      close(12)
      if(ncorr/=0) close(13)

      DEALLOCATE(ivic)
      DEALLOCATE(vpot)
      DEALLOCATE(vjas)
      DEALLOCATE(szsz)
      DEALLOCATE(tabpip)
      DEALLOCATE(iconf)

      stop
      end

!======================================================================

      subroutine neighbors(nh,ivic)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,il,ir
      INTEGER(4) ivic(nh,2)

      do i=1,nh
       ir=0
       if(i==nh) ir=nh
       il=0
       if(i==1) il=nh
       ivic(i,1)=i+1-ir        ! right
       ivic(i,2)=i-1+il        ! left
      enddo

      return
      end

      subroutine pseudo(nh,alpha,vpot,vjas)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,j
      REAL(8) pi,dist
      REAL(8) alpha
      REAL(8) vpot(nh,nh),vjas(nh,nh)

      pi=dacos(-1.d0)

      do i=1,nh
       do j=1,nh
        dist=nh/pi*dsin(pi*abs(i-j)/nh)
        vpot(i,j)=0.d0
        if(i/=j) vpot(i,j)=dlog(dist**2)
       enddo
      enddo

      do i=1,nh
       do j=1,nh
        vjas(i,j)=dexp(alpha*vpot(i,j))
       enddo
      enddo

      return
      end

      subroutine init(nh,sztot,iconf)
      implicit none
      INTEGER(4) nh,sztot,nup
      INTEGER(4) i,jr,ib
      INTEGER(4) iconf(nh)
      REAL(8) rnd

      nup=nh/2+sztot

      do i=1,nh
       iconf(i)=-1
      enddo

      ib=0
      do while(ib<nup)
       call random_number(rnd)
       jr=rnd*nh+1
       if(iconf(jr)==-1) then
        iconf(jr)=1
        ib=ib+1
       endif
      enddo

      return
```

```
      end

      subroutine upscratch(nh,iconf,vjas,tabpip)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,j
      INTEGER(4) iconf(nh)
      REAL(8) tabpip(nh),vjas(nh,nh)

! tabpip(i)=exp[\sum_k v_{i,k} (S^z_k+1/2)]
! J=exp[1/2 \sum_{i,k} v_{i,k} (S^z_i+1/2)(S^z_k+1/2)]

      do i=1,nh
       tabpip(i)=1.d0
      enddo

      do j=1,nh
       if(iconf(j)==1) then
        do i=1,nh
         tabpip(i)=tabpip(i)*vjas(i,j)
        enddo
       endif
      enddo

      return
      end

      subroutine ratiovar(i,j,nh,iconf,tabpip,vjas,ratio)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,j
      REAL(8) ratio
      INTEGER(4) iconf(nh)
      REAL(8) tabpip(nh),vjas(nh,nh)

      if(iconf(i)==-1.and.iconf(j)==1) then
       ratio=tabpip(i)/tabpip(j)*vjas(i,i)/vjas(i,j)
      elseif(iconf(i)==1.and.iconf(j)==-1) then
       ratio=tabpip(j)/tabpip(i)*vjas(i,i)/vjas(i,j)
      else
       ratio=0.d0
      endif

      return
      end
      subroutine localenergy(nh,iconf,ivic,tabpip,vjas,jperp,ener)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,jn
      REAL(8) jperp,ener
      INTEGER(4) iconf(nh),ivic(nh,2)
      REAL(8) tabpip(nh),vjas(nh,nh)

      ener=0.d0

      do i=1,nh
       jn=ivic(i,1)
       ener=ener+0.25*jperp*iconf(i)*iconf(jn)
      enddo

! the minus sign is due to the Marshall sign rule
      do i=1,nh
       jn=ivic(i,1)
       if(iconf(i)==-1.and.iconf(jn)==1) then
        ener=ener-0.5d0*tabpip(i)/tabpip(jn)*vjas(i,i)/vjas(i,jn)
       elseif(iconf(i)==1.and.iconf(jn)==-1) then
        ener=ener-0.5d0*tabpip(jn)/tabpip(i)*vjas(i,i)/vjas(i,jn)
       endif
      enddo

      return
      end

      subroutine spinspinz(nh,iconf,szsz)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,j,k
      INTEGER(4) iconf(nh)
      REAL(8) szsz(nh)

      do i=1,nh
       szsz(i)=0.d0
       do j=1,nh
        k=mod(j+i-1,nh)+1
        if(iconf(j)==iconf(k)) then
         szsz(i)=szsz(i)+0.25d0
        else
         szsz(i)=szsz(i)-0.25d0
        endif
       enddo
       szsz(i)=szsz(i)/nh
      enddo

      return
      end

      subroutine upjastrow(nh,iout,jout,iconf,tabpip,vjas)
      implicit none
      INTEGER(4) nh
      INTEGER(4) i,jn,iout,jout
      INTEGER(4) iconf(nh)
      REAL(8) tabpip(nh),vjas(nh,nh)

      if(iconf(iout)==1.and.iconf(jout)==-1) then
       do i=1,nh
        tabpip(i)=tabpip(i)/vjas(i,iout)
        tabpip(i)=tabpip(i)*vjas(i,jout)
       enddo
       iconf(iout)=-1
       iconf(jout)=1
      elseif(iconf(iout)==-1.and.iconf(jout)==1) then
       do i=1,nh
        tabpip(i)=tabpip(i)*vjas(i,iout)
        tabpip(i)=tabpip(i)/vjas(i,jout)
       enddo
       iconf(iout)=1
       iconf(jout)=-1
      else
       write(6,*) 'Wrong updating'
       stop
      endif

      return
      end
```

With input file:

```
&lattice
nh=20
/
&parameters
jperp=1.d0
sztot=0
/
&wavefunction
```

```
alpha=1.d0
/
&montecarlo
iseed(1)=12345
iseed(2)=54321
iseed(3)=34567
iseed(4)=76543
iseed(5)=56789
iseed(6)=98765
iseed(7)=13579
iseed(8)=97531
ngen=1000000
nbra=100
nscra=100
ncorr=0
/
```

The codes to compute energy and correlations with the binning technique are:

```
      program readene
      implicit none
      INTEGER(4) i
      INTEGER(4) lbin,ibinit,kt,ibin,nmis
      REAL(8) ek,wk,ebin,ebin2,wbin,de

      write(6,*) 'lbin,ndrop'
      read(5,*) lbin,ibinit

      kt=0
      ibin=0

      ek=0.d0
      ebin=0.d0
      ebin2=0.d0
      wk=0.d0
      wbin=0.d0

      do while(kt.ge.0)
       kt=kt+1
       read(12,end=100) i,de
       ek=ek+de
       wk=wk+1.d0
       if(mod(kt,lbin).eq.0) then
        ibin=ibin+1
```

```
          if(ibin.ge.ibinit) then
           ebin=ebin+ek
           wbin=wbin+wk
           ebin2=ebin2+ek**2/wk
          endif
          write(20,*) ek,wk
          ek=0.d0
          wk=0.d0
         endif
        enddo

100     continue
        nmis=ibin-ibinit+1
        ebin=ebin/wbin
        ebin2=dsqrt(dabs(ebin2/wbin-ebin**2))
        ebin2=ebin2/dsqrt(dfloat(nmis))
        write(6,*) ' Independent bins ',nmis,'of lenght ',lbin
        write(6,*)
        write(6,*) ' Energy'
        write(6,*) ebin,ebin2
        write(6,*)

        stop
        end

        program readcorr
        implicit none
        INTEGER(4) nh
        INTEGER(4) i,j
        INTEGER(4) lbin,ibinit,kt,ibin,nmis
        REAL(8) wk,wbin
        REAL(8), dimension(:), allocatable :: de,ek
        REAL(8), dimension(:), allocatable :: ebin,ebin2

        write(6,*) 'lbin,ndrop,size'
        read(5,*) lbin,ibinit,nh

        ALLOCATE(de(nh))
        ALLOCATE(ek(nh))
        ALLOCATE(ebin(nh))
        ALLOCATE(ebin2(nh))

        kt=0
        ibin=0

        do j=1,nh
         ek(j)=0.d0
         ebin(j)=0.d0
         ebin2(j)=0.d0
        enddo
        wk=0.d0
        wbin=0.d0

        do while(kt.ge.0)
         kt=kt+1
         read(13,end=100) i,(de(j),j=1,nh)
         do j=1,nh
          ek(j)=ek(j)+de(j)
         enddo
         wk=wk+1.d0
         if(mod(kt,lbin).eq.0) then
          ibin=ibin+1
          if(ibin.ge.ibinit) then
           do j=1,nh
            ebin(j)=ebin(j)+ek(j)
           enddo
           wbin=wbin+wk
           do j=1,nh
            ebin2(j)=ebin2(j)+ek(j)**2/wk
           enddo
          endif
          write(20,*) (ek(j),j=1,nh),wk
          do j=1,nh
           ek(j)=0.d0
          enddo
          wk=0.d0
         endif
        enddo

  100    continue
        nmis=ibin-ibinit+1
        do j=1,nh
         ebin(j)=ebin(j)/wbin
         ebin2(j)=dsqrt(dabs(ebin2(j)/wbin-ebin(j)**2))
         ebin2(j)=ebin2(j)/dsqrt(dfloat(nmis))
        enddo
        write(6,*) ' Independent bins ',nmis,'of lenght ',lbin
        write(6,*)
        write(6,*) ' Correlations'
        do j=1,nh
         write(6,*) j,ebin(j),ebin2(j)
        enddo
```

```
write(6,*)

DEALLOCATE(de)
DEALLOCATE(ek)
DEALLOCATE(ebin)
DEALLOCATE(ebin2)

stop
end
```