HW9 Anisotropic Heisenberg Antiferromagnet

Fei Ding

December 18, 2009

(i)

$$|\varphi_{n+1}>= H|\varphi_n> -a_n|\varphi_n> -b_n^2|\varphi_{n-1}>$$

$$a_n =< \varphi_n|H|\varphi_n > / < \varphi_n|\varphi_n >$$

$$b_n^2 =< \varphi_n|\varphi_n > / < \varphi_{n-1}|\varphi_{n-1} >$$

Multiply the first equation by $< \varphi_n|$, we get

$$\begin{aligned} < \varphi_n|\varphi_{n+1} > &= < \varphi_n|H|\varphi_n > -a_n < \varphi_n|\varphi_n > -b_n^2 < \varphi_n|\varphi_{n-1} > \\ &= < \varphi_n|H|\varphi_n > - < \varphi_n|H|\varphi_n > / < \varphi_n|\varphi_n > \cdot < \varphi_n|\varphi_n > -b_n^2 < \varphi_n|\varphi_{n-1} > \\ &= < \varphi_n|H|\varphi_n > - < \varphi_n|H|\varphi_n > -b_n^2 < \varphi_n|\varphi_{n-1} > \\ &= -b_n^2 < \varphi_n|\varphi_{n-1} > \end{aligned}$$

Associated with the initial value $|\varphi_{-1}>= 0$, we get (suppose all the $|\varphi_n>$ are real),

$$< \varphi_n|\varphi_{n+1} >= -b_n^2 < \varphi_{n-1}|\varphi_n >= -b_n^2 b_{n-1}^2 < \varphi_{n-2}|\varphi_{n-1} >= \cdots = \prod_{i=0}^{n} b_n^2 < \varphi_{-1}|\varphi_0 >= 0$$

(ii) For $L = 4$, $g = 1$ case, I got $E_0/JL = -0.5$, $E_1/JL = -0.25$ after about three iterations, which is good. However, there is one problem rising up. Although it's trivial to get the ground state energy, it's not quite easy for me to find the first excited state energy. In the $L = 4$ case, I actually know the analytical result so I was just looking for that value although it only appeared steadily twice and crash down quickly to the value very close to E_0. I guess this is because of the round off error stuff we talked about in HW2, problem 4. My first guess is that just like E_0/JL, the value of E_0/JL for infinite case is -0.44, which is not far away from $L = 4$ case, the E_1/JL should be around -0.25 anyway. However, for the $L > 4$ case, it's really difficult for me to figure out a value for E_1/JL which is both close to -0.25 and steady. It can be that the speed of E_1 collapse to E_0 is larger than the speed of E_1 itself converges. However, I would rather pick up the value which looks steady to be the $E1$ value. Thus basically this is what I did for the 4th and 5th question. My guess is that as L increases, there is higher possibility to get degenerate states, so the first excited state energy(actually this is just the second smallest eigenvalue of the Hamiltonian matrix) gets closer to the ground state energy(the smallest eigenvalue of the matrix). What I am saying is that finally we would get $E1 \approx E0$.

(iii) The plot of E_0/JL vs. g for $L = 12$ is shown in Figure 1, I did 40 iterations for each g, which is enough for L=12 case.

(iv) As I said in the 2nd question, I just pick up the $E1$ which looks steady. I changed L manually because I have to keep an eye to see how many iterations I need. Of course I can jazz up the program to let it

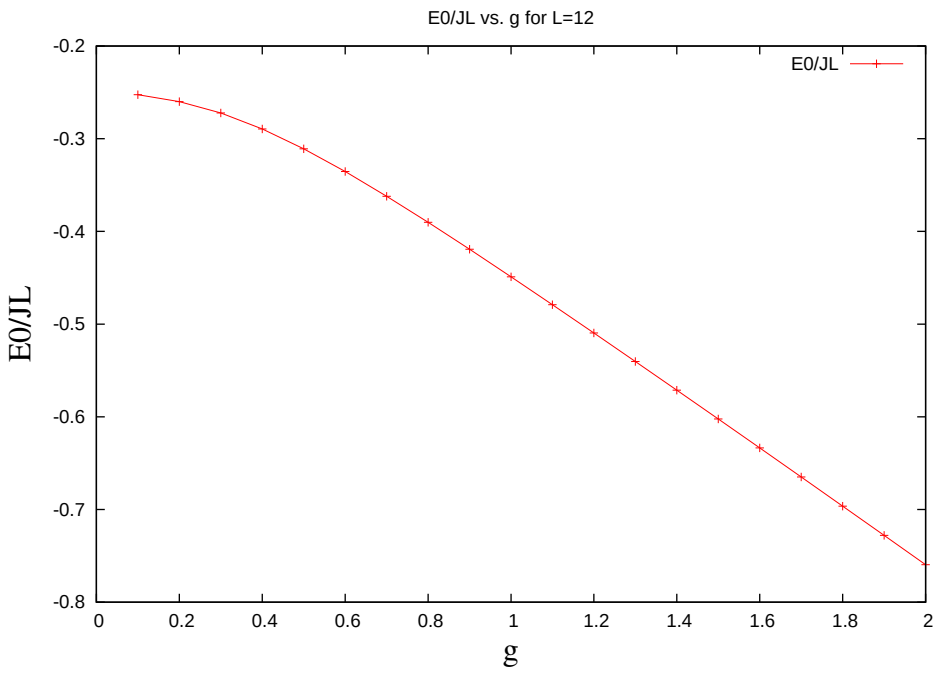

Figure 1: E_0/JL vs. g

change L by itself but just short of time. I stopped at $L = 20$ because it took the computer about 40 min to get 35 iterations at which point the result didn't converge yet. (By the way, can I know how you made your program run $L = 20$ so quickly?) Here's the data and the plot(Figure 2). I guess the infinite case E_0/JL should be around -0.44, but I cannot get it as accurate as 5 or 6 digits. And the E_1/JL should finally be very close to E_0/JL.

L	$L_0(S_z = 0$ states)	E_0/JL	$E1/JL$	$(E_1 - E_0)/JL$	Iteration
4	6	-0.500000	-0.250000	0.250000	3
6	20	-0.467129	-0.353005	0.114124	9
8	70	-0.456386	-0.391052	6.53343×10^{-2}	13
10	252	-0.451544	-0.409220	4.23239×10^{-2}	18
12	924	-0.448949	-0.419295	2.96539×10^{-2}	24
14	3432	-0.447396	-0.425460	2.19361×10^{-2}	28
16	12870	-0.446393	-0.429506	1.68868×10^{-2}	34
18	48620	-0.445708	-0.432306	1.34027×10^{-2}	40

(v) Again, the same issue about $E1$ as I said before. I think it means that for bulk limit, the energy may no longer be discrete but continuous, in the case the $E1$s I wrote here are correct. So I think for the bulk limit $(E_1 - E_0)/JL$ should be very close to zero. The plot is shown below in Figure 3.

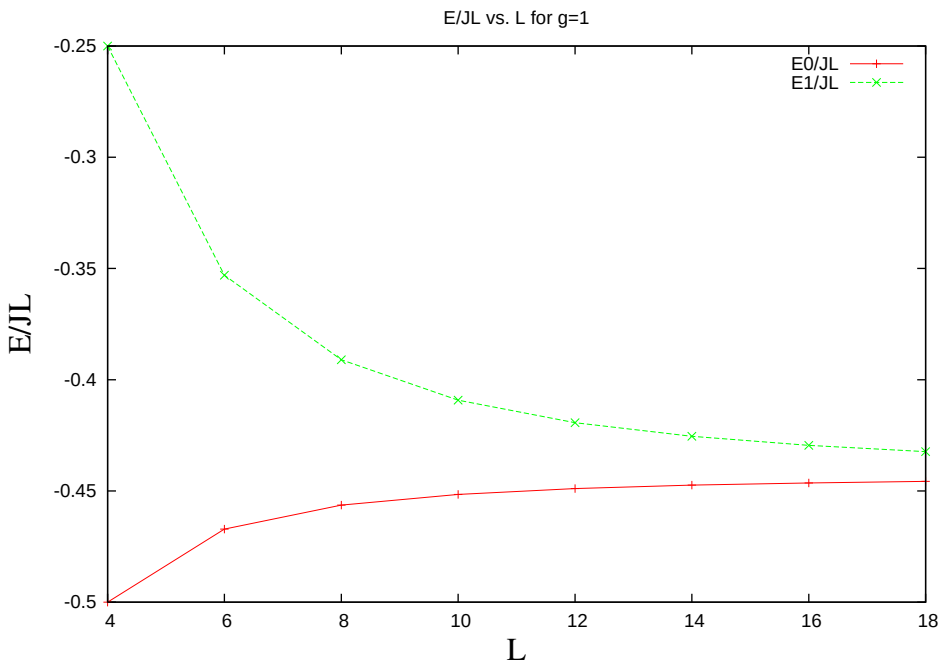

Figure 2: E/JL vs. L

A Codes

Note: The code looks ugly because I type in a lot of print statement to debug the code. I just comment them out.

```
program main
IMPLICIT NONE
integer,parameter::L=12,N=4096,L0=924,m=50 !
# of particles,# of all possible states, # of Sz=0 states
integer::i,j
double precision::g=1.0d0
double precision,dimension(L0)::Hphi,phi,phiprev,phinew
double precision::phiHphi,phi2,phiprev2
integer,dimension(L0)::relable

!the parameters in the tridiagonal form of H in phi basis
double precision,dimension(0:m)::a,aa
double precision,dimension(0:m)::b,bb
double precision,dimension(m+1,m+1)::z
integer::flag
z=0d0


call basis(L,N,L0,relable)

phiprev=0d0
do i=1,L0
   if (i==1.or.i==2) then
      phi(i)=1d0/dsqrt(2d0)
   else
      phi(i)=0d0
end if
end do
```

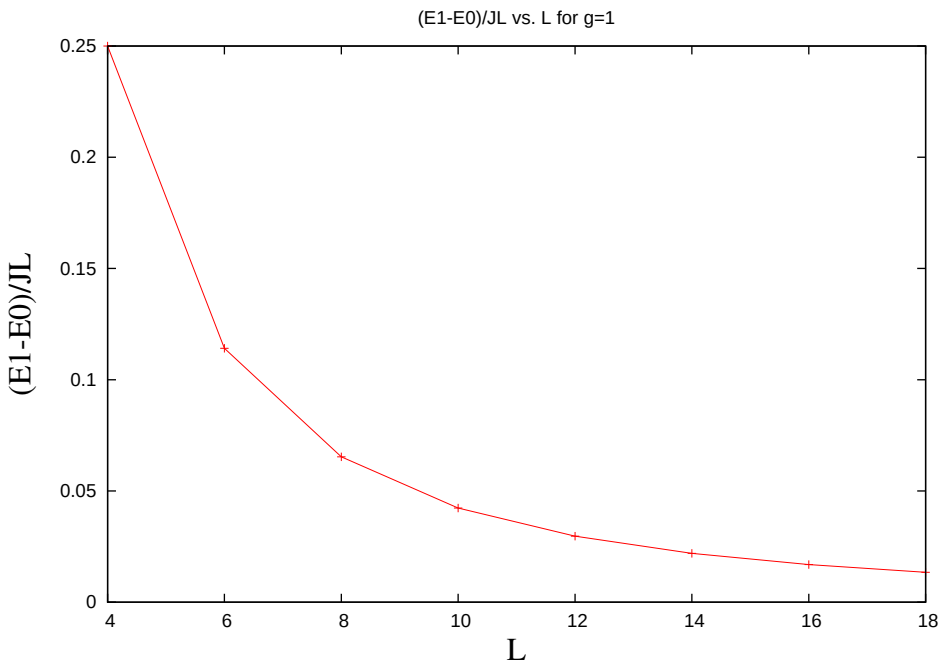

Figure 3: $(E_1 - E_0)/JL$ vs. L

```
a=0d0
b=0d0
print *,'g=',g
phiprev2=0d0
do j=0,m
!loop of phi_j to get a_j, b_j until |phi_j|=0

 phiHphi=0d0
   phi2=0d0
   call Hamiltonian(relable,L0,L,g,phi,Hphi)
!   print *,'j=',j
!   print *,'after call,phi=',phi,'Hphi=',Hphi
   do i=1,L0!loop inside phi to calculate norm
      !call Hamiltonian(relable,L0,L,g,phi,Hphi)
      phiHphi=phiHphi+phi(i)*Hphi(i)
!   print *,'i=',i,phi(i)
   phi2=phi2+phi(i)*phi(i)
!   print *,'i=',i,phi(i),Hphi(i)
!   print *,'phiHphi=',phiHphi,'phi2=',phi2

   !IF (phi2==0d0) EXIT
   end do
!   print *,'j=',j,'phiHphi=',phiHphi,'phi2=',phi2
   a(j)=phiHphi/phi2
   if (j==0) then
      b(j)=0d0!b_0=0
   else
      b(j)=phi2/phiprev2
   end if
!print *,'j=',j,'phi2=',phi2,'phiprev2=',phiprev2
```

```
   phiprev2=phi2
   phinew=Hphi-a(j)*phi-b(j)*phiprev
!   print *,'phinew=',phinew
!   print *,'phi=',phi
   phiprev=phi
   phi=phinew
!   print *,'j=',j
!   print *,'phi=',phi
!   print *,'phinew=',phinew
!   j=j+1

if (j.NE.0) then
   aa=a
   bb=dsqrt(b)
   call imtql2(m+1,m+1,aa,bb,z,flag)
print *,'j=',j,'flag=',flag,'E0/JL=',aa(0)/dble(L),'E1/JL=',aa(1)/dble(L)
print *,'delta=',(aa(1)-aa(0))/dble(L)
!print *,aa(4)
!print *, 'aa=',aa
end if
end do
!print *,'g=',g,'E0/JL=',aa(0)/dble(L)


!print *,'a=',a
!print *,'b=',b
pause
end program

INCLUDE 'imtql2.f'
INCLUDE 'pythag.f'
!go through all possible states to find out Sz=0 states, and relable them



subroutine basis(L,N,L0,relable)
integer::N,L,L0
integer,dimension(L0)::relable
integer,dimension(L)::state

integer::i
integer::sc,rsc! state code and relabel state code
integer::Sz !total Sz
External::dec2bin
!integer:: r
rsc=1
do sc=0,N-1
   call dec2bin(sc,L,state)
   Sz=0
   do i=1,L
      Sz=Sz+state(i)
   end do !loop of calculating Sz
   if (Sz==L/2) then
      relable(rsc)=sc !rsc is the index of relable array;sc is the content of relable array
      rsc=rsc+1
   end if
end do !loop of going thtough all states
```

```
do i=1,L0
!   print *,'relabel=', relable(i)
end do
end subroutine basis

subroutine Hamiltonian(relable,L0,L,g,phi,Hphi)
integer::L0,L
double precision::g
double precision,dimension(L0)::phi,Hphi,Temphi
integer,dimension(L0)::relable

integer::i
integer,dimension(L)::state,flip,pl
integer::sc,rsc! state code and relabel state code
integer::fsc,frsc! flip state code and relabel flip state code
double precision::h0
external::dec2bin,bin2dec,search
!pl is the PBC next
do i=1,L
   if (i==L) then
      pl(i)=1
   else
      pl(i)=i+1
   end if
end do
!print *,'pl=',pl

Hphi=0d0
do rsc=1,L0
   Temphi=0d0
   if(phi(rsc).NE.0) then
     sc=relable(rsc)
!     print *,'rsc=',rsc,'sc=',sc
     call dec2bin(sc,L,state) !decode state
!print *,state
     h0=0d0
     do i=1,L
        flip=state
 !       print *,'flip=',flip

        if (state(i)==state(pl(i))) then
           h0=h0+1d0
! print *,'i=',i,'h0=',h0
        else
!print *,'i=',i,'h0=',h0

           h0=h0-1d0
           flip(i)=state(pl(i))
           flip(pl(i))=state(i)
!print *,state
!print *,flip
           call bin2dec(flip,L,fsc)
!print *,fsc
           call search(relable,L0,fsc,frsc)
!print *,frsc
!Hphi(frsc)=Hphi(frsc)+1
           Temphi(frsc)=0.5d0*g
        end if
```

```
     end do!loop of different spins in one state
!     print *,'h0=',h0
     Temphi(rsc)=0.25d0*h0
    ! Hphi(rsc)=Hphi(rsc)+h0
!print *,'phi=',phi
!print *,'Temphi=',Temphi
     Hphi=phi(rsc)*Temphi+Hphi
   end if

end do

!print *,phi
!print *,Hphi
end subroutine Hamiltonian


subroutine search(relable,L0,sc,rsc)
IMPLICIT NONE
integer::i,L0,rsc,sc
integer,dimension(L0)::relable
integer::TOP,BOT,MID,FIND

FIND=0
TOP=1
BOT=L0
do  while(TOP<=BOT.AND.FIND==0)
   MID=(TOP+BOT)/2
   If(sc==relable(MID)) then
      FIND=1
      rsc=MID
   else if(sc<relable(MID)) then
      BOT=MID-1
   else
      TOP=MID+1
   end if
end do

if (find==0) then
    print*,'Error!'
end if
end subroutine search


subroutine dec2bin(dec,L,bin) !In:dec,L(the dimension of array bin); Out:bin
IMPLICIT NONE
integer:: i,l,dec,decTemp
integer,dimension (L):: bin !binary presentation of the basis state
decTemp=dec
do i=L,1,-1
   bin(i)=mod(decTemp,2)
   decTemp=decTemp/2
!   print *,'bin=',bin(i)
end do
end subroutine dec2bin


subroutine bin2dec(bin,L,dec) !In:bin,L(the dimension of array bin); Out:dec
IMPLICIT NONE
```

```
integer,dimension(L):: bin
integer::i,L,dec
integer::decTemp
decTemp=0
do i=L,1,-1
   decTemp=decTemp+bin(i)*(2**(L-i))
end do
   dec=decTemp
end subroutine bin2dec
```