Energy Spectrum of Quantum Spin Chains

Michael Papasimeon
10 September 1997

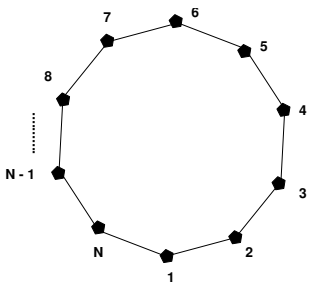
I. Aims

The main aims are to:

- Write a computer program which represents the complete set of states of a quantum spin chain of size N.
- Modify the program to compute the Hamiltonian matrix of the quantum spin chain
- Use package routines to tri-diagonalise and then to diagonalise the Hamiltonian matrix which will give us the eigenvalues of the matrix.
- The eigenvalues are the energy eigenstates of the quantum spin chain obtained from the Schrodinger equation $H|\Psi\rangle = E|\Psi\rangle$. This gives us the energy spectrum of the quantum spin chain for a given N.
- Investigate the energy spectrum of the quantum spin chain if it is placed in a magnetic field oriented along it's z-axis with a magnetic field strength of $B_z\mu$.

II. Background

The diagram below represents a quantum spin chain of N particles. Each particle's z-component spin is either in a spin up $\left(+\frac{1}{2}\right)$ state or a spin down $\left(-\frac{1}{2}\right)$ state.



Each particle interacts with it's neighbour. The interaction can be described with the Heisenberg Hamiltonian H,

$$H = \sum_{i=1}^{N} \vec{S}(i) \cdot \vec{S}(i+1)$$

where i is the particle site, and $\vec{S}(i)$ is a vector of spin operators at site i, such that $\vec{S}(i) = (S_x(i), S_y(i), S_z(i))i$ The chain can be in a number of different spin states at any one time. There are 2^N possible states for N particles. The complete set of states is given by:

$$\{|\phi_n\rangle\} = \{|S_z(1)S_z(2)S_z(3).....S_z(N-1)S_z(N)\rangle\}$$

For example for a chain with $N = 2$, the complete set of states is given by:

$$\{|\phi_n\rangle\} = \{|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\downarrow\downarrow\rangle\}$$

Using this information we can solve the Schrodinger eigenvalue equation of the system:

$$H|\psi\rangle = E|\psi\rangle$$

This can be represented as a matrix equation:

$$Hx = E_n x$$

where E_n are the energy levels of the system, x_n are the eigenvectors, and H is the Heisenberg Hamiltonian matrix. Therefore:

$$x_n = \langle\phi_n|\psi\rangle$$

and each matrix element H_{nm} is given by:

$$H_{nm} = \langle\phi_n|H|\phi_m\rangle$$

The Heisenberg Hamiltonian can be expanded as follows:

$$H = \sum_{i=1}^{N} \vec{S}(i) \cdot \vec{S}(i+1)$$

$$H = \sum_{i=1}^{N} [S_x(i)S_x(i+1) + S_y(i)S_y(i+1) + S_z(i)S_z(i+1)]$$

Using the raising and lowering operators:

$$S_\pm = S_x(i) \pm iS_y(i)$$

the Hamiltonian H can be written in terms of two components such that:

$$H = \sum_{i=1}^{N} [H_z(i) + H_f(i)]$$

Where:

$$H_z(i) = S_z(i)S_z(i+1)$$

$$H_f(i) = \frac{1}{2}\left[S_+(i)S_-(i+1) + S_-(i)S_+(i+1)\right]$$

III. Method

A Fortran program (`spin.f` in Appendix A)was written to represent the complete set of states of a quantum spin chain. The binary representation of 64-bit integers was used to represent a possible state of a quantum spin chain. A given integer represents one possible state which the chain could be in. The binary representation of each integer is made up of 0's and 1's, where 0 represents spin down $-\frac{1}{2}$ and 1 represents spin up $+\frac{1}{2}$. Each bit position represents the next site in the chain starting with the least significant bit as the first site. Table I shows the representation of the complete set of states using this method for a chain of size $N = 4$.

TABLE I
BINARY REPRESENTATION OF COMPLETE SET OF STATES $\{|\phi_i\rangle\}$ FOR $N = 4$

State $\|\phi_i\rangle$	2^3	2^2	2^1	2^0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	1
5	0	1	0	0
6	0	1	0	1
7	0	1	1	0
8	0	1	1	1
9	1	0	0	0
10	1	0	0	1
11	1	0	1	0
12	1	0	1	1
13	1	1	0	0
14	1	1	0	1
15	1	1	1	0
16	1	1	1	1

TABLE III
ERROR CODES WHEN DIAGONALISING MATRICES FOR DIFFERENT N

N	`ierr`
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	6
10	2

Once this is done functions to calculate the matrix elements H_{mn} of the Hamiltonian matrix were written. Functions are needed to calculate the following quantaties:

- $\langle\phi_j|H_z|\phi_i\rangle$
- $\langle\phi_j|H_f|\phi_i\rangle$
- For the second part the contribution to each matrix element from the magnetic field $\langle\phi_j|H_m|\phi_i\rangle$

Once the Hamiltonian matrix for the particular spin chain has been calculated, package Fortran subroutines are used to tri-diagonalise and then diagonalise the matrix. The elements along the diagonal of the resulting matrix give the energy eigenvalues of the system.

IV. RESULTS

Energy Eigenvalues for Various Chain Sizes

Table **??** shows the results of running the program for values of ranging from $N = 2$ to $N = 10$. The results in the table are E_i/N where i is the ith energy eigenstate and N is the size of the quantum spin chain.

Table III shows the error code `ierr` given from the `imtql1` subroutine for the different values of N. The error code is 0 if no error occured and a positive integer if an error occured. The value is set j if the jth eigenvalue has not being determined after 30 iterations.

TABLE IV
ENERGY LEVELS E_0 FOR VARYING $B_z\mu$ (`1.dat`)

$B_z\mu$	E_0
0	-2.75391005
1	-3.2783723
2	-6.83931233
5	-17.976674
10	-36.7141831
20	-74.2665118
50	-186.98981
100	-374.884865
500	-1878.08728
1000	-3757.09559

TABLE V
ENERGY LEVELS E_1 FOR VARYING $B_z\mu$ (`2.dat`)

$B_z\mu$	E_1
0	-2.04416367
1	-2.56881924
2	-2.53377674
5	-5.17989513
10	-10.8286411
20	-22.1589593
50	-56.1740549
100	-112.873705
500	-566.484799
1000	-1133.50039

Spin Chains in the presence of a Magnetic Field of strength $B_z\mu$

Tables IV– IX show the energy eigenvalues for a chain of $N = 6$, for different values of the magnetic field strength $B_z\mu$. Each table shows a particular energy level. For example Table IV shows the results for the lowest energy level E_0.

The data in Tables IV– IX have been plotted in Figure 1. The lines `1.dat` – `6.dat` correspond to the data in the different tables. Each lines in the plot corresponds to one of the six energy eigenvalues of the $N = 6$ quantum spin chain.

V. DISCUSSION

Given the fact that there were some errors in the diagonalisation procedure as shown by Table 3 for $N = 8, 9, 10$, the results shown in Table 2 for the energy levels of spin chains of this size are unlikely to be correct.

TABLE VI
ENERGY LEVELS E_2 FOR VARYING $B_z\mu$ (`3.dat`)

$B_z\mu$	E_2
0	-2.04416367
1	-1.92139214
2	-1.89004066
5	-3.20982437
10	-6.00296934
20	-11.7495711
50	-29.062476
100	-57.9371933
500	-288.967431
1000	-577.759148

TABLE II
ENERGY EIGENVALUES E_i/N FOR DIFFERENT SIZED QUANTUM SPIN CHAINS

i	N=2	N=3	N=4	N=5	N=6	N=7	N=8	N=9	N=10
0	-0.75	-0.583	-1.140	-1.534	-2.754	-4.411	1.860E-18	0.028	-0.050
1	0.25	-0.583	-0.500	-1.534	-2.044	-4.411	1.315E-17	0.028	-0.050
2		-0.083	-0.500	-0.531	-2.044	-2.811	2.862E-17	0.028	-0.050
3			-0.161	-0.531	-0.634	-2.811	8.822E-18	0.028	-0.050
4				-0.150	-0.634	-0.823	1.713E-18	0.028	-0.050
5					-0.231	-0.823	-1.02E-17	0.028	-0.050
6						-0.179	-3.65E-19	0.028	-0.050
7							-1.58E-19	0.028	-0.050
8								0.028	-0.050
9									-0.050

TABLE VII
ENERGY LEVELS E_3 FOR VARYING $B_z\mu$ (`4.dat`)

$B_z\mu$	E_4
0	-.63379594
1	-1.70564112
2	-1.84952222
5	-2.34410019
10	-4.1755527
20	-8.63991335
50	-22.0554154
100	-44.422037
500	-223.368376
1000	-447.052966

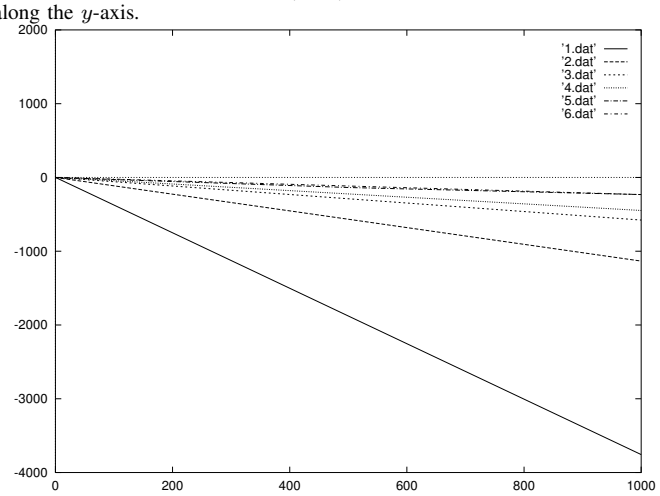
TABLE VIII
ENERGY LEVELS E_4 FOR VARYING $B_z\mu$ (`5.dat`)

$B_z\mu$	E_4
0	-.63379594
1	-.752476436
2	-1.66090753
5	-1.95855966
10	-2.98392009
20	-5.64385606
50	-13.7786674
100	-27.3685291
500	-136.135355
1000	-232.535479

TABLE IX
ENERGY LEVELS E_5 FOR VARYING $B_z\mu$ (`6.dat`)

$B_z\mu$	E_5
0	-.230940787
1	-.596153367
2	-.883832797
5	-1.84820487
10	-2.29196661
20	-4.59813971
50	-11.5734487
100	-23.2022424
500	-116.238876
1000	-232.535479

Fig. 1. Magnetic Field Strength ($B_z\mu$) along the x-axis and Energy Level along the y-axis.



This leaves us with energy levels for spin chains of size $N = 2$ to $N = 7$. For the ground state energy we expect the following result for $N \to \infty$:

$$\frac{E_0}{N} = -\ln(2) + \frac{1}{4} \simeq -0.443$$

This does not compare favourably with the results in Table 2, because as N is increased E_0/N is getting smaller with $E_0/7 = -4.411$.

For $N = 4$, the expected eigenvalues are:

$$\begin{aligned} \frac{E_0}{N} &= -0.5 \\ \frac{E_1}{N} &= -0.25 \\ \frac{E_2}{N} &= -0.25 \\ \frac{E_3}{N} &= -0.25 \end{aligned}$$

Comparing to table 2, we see that the eigenvalue of -0.5 is the only one in agreement. The lack of agreement indicates an error in the program. However for the simple case of $N = 2$, the Hamiltonian matrix produced by the program for the set

of states described in table 1 is as follows:

$$H_{N=2} = \begin{pmatrix} -0.5 & 1 & 0 & 0 \\ 1 & -0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}$$

This result is as expected, which suggests that the routines `tred1` and/or `imtql1` may not be working as expected. However, due to time considerations, the problem was not traced to any part of the code in particular.

For the case when the quantum spin chain is in a magnetic field, we get the results shown in Plot 1. The plots follow the form of the expected output where:

$$E(\mu B_z) = -MB_z = -\mu B_z(N(\uparrow) - N(\downarrow))$$

The gradient of the plots is determined by the factor $(N(\uparrow) - N(\downarrow))$. For the six energy eigenvalues $(N(\uparrow) - N(\downarrow)) > 0$. This indicates that for the $N = 6$ spin chain, we have more particles in the spin up state than in the spin down state. The ground state energy has the largest gradient in Plot 1, indicating that there are more particles in the spin state than in the spin down state. This however decreases as the we go to higher energy states, making the gradient in the plot smaller.

APPENDIX
CODE LISTING: SPIN.F

```
c-------------------------------------------------------------------------------

      program main
            implicit none
               integer*4 N
               integer*4 row, col, ierr, i
               parameter(N=4)
            real*8 Bzmu
            real*8 Hz, Hf, Hm
               real*8 ham(2**N, 2**N)
               real*8 d(2**N), e(2**N), e2(2**N)

            write(*,*)'Enter Bz*mu'
            read(*,*) Bzmu

               do row = 1, 2**N, +1
                      do col = 1, 2**N, +1
                             if (row .eq. col) then
                                    ham(row,col) = Hz(col,N)
                             else
                                    ham(row,col) = 0.0d0
                             end if
                      ham(row, col) = ham(row,col) + Hf(row,col,N)
     > - Hm(col,N,Bzmu)
                      end do
               end do

            call tred1(2**N, 2**N, ham, d, e, e2)
            call imtql1(2**N, d, e, ierr)

               do row = 1, 2**N, +1
                      write(*,111) (ham(row, col), col = 1, 2**N)
111     format(f5.1,f5.1,f5.1,f5.1,f5.1,f5.1,f5.1,f5.1,
     >    f5.1,f5.1,f5.1,f5.1,f5.1,f5.1,f5.1,f5.1)
               end do

            write(*,*)'Eigenvalues', ierr

            do i = 1, N, +1
                 write(*,*)'i d[i]/N = ',i, d(i)/dfloat(N)
               end do
         end

c-------------------------------------------------------------------------------

         double precision function Hz(state,N)
                implicit none
                integer*4 state, N, site
                real*8 sum, Sz

                sum = 0.0d0

                do site = 0, N - 1, +1
                       if (site .eq. N - 1) then
                              sum = sum + Sz(state, site)*Sz(state, 0)
                       else
                              sum = sum + Sz(state, site)*Sz(state, site + 1)
                       endif
                end do

                Hz = sum
                return
         end


```

```
c-------------------------------------------------------------------------

      double precision function Hf(row, col, N)
            implicit none
            integer*4 i, row, col, N, site, next
            real*8 sum, Sz

            sum = 0.0d0

            do i = 0, N - 1, +1
                  site = i
                  if (site .eq. N - 1) then
                        next = 0
                  else
                        next = i + 1
                  endif

                  if ( Sz(row,site)*Sz(row,next) .lt. 0.0d0 ) then
                        if ((Sz(col,site) .eq. -Sz(row,site)) .and.
     > (Sz(col,next) .eq. -Sz(row,next)) ) then
                              sum = sum + 0.5d0
                        end if
                  endif
            end do

            Hf = sum
            return
      end

c-------------------------------------------------------------------------

      double precision function Hm(state, N, Bzmu)
            implicit none
            integer*4 state, N, i
            real*8 Bzmu, sum, Sz

            sum = 0.0d0

            do i = 0, N - 1, +1
                  sum = sum + 0.5*Bzmu*Sz(state,i)
            end do

            Hm = sum
            return
      end

c-------------------------------------------------------------------------

        double precision function Sz(state, site)
              implicit none
              integer*4 state, site
              real*8 spin

              spin = and(state, 2**site)
              spin = spin/(2**site)

              if (spin .gt. 0.0d0) then
                    Sz = 0.5d0
              else
                    Sz = -0.5d0
              endif

              return
        end

c-------------------------------------------------------------------------
```

```
      subroutine tred1(nm,n,a,d,e,e2)
c
      integer i,j,k,l,n,ii,nm,jp1
      double precision a(nm,n),d(n),e(n),e2(n)
      double precision f,g,h,scale
c
c this subroutine is a translation of the algol procedure tred1,
c num. math. 11, 181-195(1968) by martin, reinsch, and wilkinson.
c handbook for auto. comp., vol.ii-linear algebra, 212-226(1971).
c
c this subroutine reduces a real symmetric matrix
c to a symmetric tridiagonal matrix using
c orthogonal similarity transformations.
c
c on input
c
c nm must be set to the row dimension of two-dimensional
c array parameters as declared in the calling program
c dimension statement.
c
c n is the order of the matrix.
c
c a contains the real symmetric input matrix. only the
c lower triangle of the matrix need be supplied.
c
c on output
c
c a contains information about the orthogonal trans-
c formations used in the reduction in its strict lower
c triangle. the full upper triangle of a is unaltered.
c
c d contains the diagonal elements of the tridiagonal matrix.
c
c e contains the subdiagonal elements of the tridiagonal
c matrix in its last n-1 positions. e(1) is set to zero.
c
c e2 contains the squares of the corresponding elements of e.
c e2 may coincide with e if the squares are not needed.
c
c questions and comments should be directed to burton s. garbow,
c mathematics and computer science div, argonne national laboratory
c
c this version dated august 1983.
c
c ------------------------------------------------------------------
c
      do 100 i = 1, n
         d(i) = a(n,i)
         a(n,i) = a(i,i)
  100 continue
c ..... for i=n step -1 until 1 do -- .....
      do 300 ii = 1, n
         i = n + 1 - ii
         l = i - 1
         h = 0.0d0
         scale = 0.0d0
         if (l .lt. 1) go to 130
c ..... scale row (algol tol then not needed) .....
         do 120 k = 1, l
  120 scale = scale + dabs(d(k))
c
         if (scale .ne. 0.0d0) go to 140
c
         do 125 j = 1, l
            d(j) = a(l,j)
            a(l,j) = a(i,j)
            a(i,j) = 0.0d0
  125 continue
```

```
c
  130 e(i) = 0.0d0
         e2(i) = 0.0d0
         go to 300
c
  140 do 150 k = 1, l
            d(k) = d(k) / scale
            h = h + d(k) * d(k)
  150 continue
c
         e2(i) = scale * scale * h
         f = d(l)
         g = -dsign(dsqrt(h),f)
         e(i) = scale * g
         h = h - f * g
         d(l) = f - g
         if (l .eq. 1) go to 285
c ..... form a*u .....
         do 170 j = 1, l
  170 e(j) = 0.0d0
c
         do 240 j = 1, l
            f = d(j)
            g = e(j) + a(j,j) * f
            jp1 = j + 1
            if (l .lt. jp1) go to 220
c
            do 200 k = jp1, l
               g = g + a(k,j) * d(k)
               e(k) = e(k) + a(k,j) * f
  200 continue
c
  220 e(j) = g
  240 continue
c ..... form p .....
         f = 0.0d0
c
         do 245 j = 1, l
            e(j) = e(j) / h
            f = f + e(j) * d(j)
  245 continue
c
         h = f / (h + h)
c ..... form q .....
         do 250 j = 1, l
  250 e(j) = e(j) - h * d(j)
c ..... form reduced a .....
         do 280 j = 1, l
            f = d(j)
            g = e(j)
c
            do 260 k = j, l
  260 a(k,j) = a(k,j) - f * e(k) - g * d(k)
c
  280 continue
c
  285 do 290 j = 1, l
            f = d(j)
            d(j) = a(l,j)
            a(l,j) = a(i,j)
            a(i,j) = f * scale
  290 continue
c
  300 continue
c
      return
      end

```

```
c----------------------------------------------------------------------------

c ----------------------------------------------------
       subroutine imtql1(n,d,e,ierr)
c
      integer i,j,l,m,n,ii,mml,ierr
      double precision d(n),e(n)
      double precision b,c,f,g,p,r,s,tst1,tst2,pythag
c
c this subroutine is a translation of the algol procedure imtql1,
c num. math. 12, 377-383(1968) by martin and wilkinson,
c as modified in num. math. 15, 450(1970) by dubrulle.
c handbook for auto. comp., vol.ii-linear algebra, 241-248(1971).
c
c this subroutine finds the eigenvalues of a symmetric
c tridiagonal matrix by the implicit ql method.
c
c on input
c
c n is the order of the matrix.
c
c d contains the diagonal elements of the input matrix.
c
c e contains the subdiagonal elements of the input matrix
c in its last n-1 positions. e(1) is arbitrary.
c
c on output
c
c d contains the eigenvalues in ascending order. if an
c error exit is made, the eigenvalues are correct and
c ordered for indices 1,2,...ierr-1, but may not be
c the smallest eigenvalues.
c
c e has been destroyed.
c
c ierr is set to
c zero for normal return,
c j if the j-th eigenvalue has not been
c determined after 30 iterations.
c
c calls pythag for dsqrt(a*a + b*b) .
c
c questions and comments should be directed to burton s. garbow,
c mathematics and computer science div, argonne national laboratory
c
c this version dated august 1983.
c
c ----------------------------------------------------------------------
c
      ierr = 0
      if (n .eq. 1) go to 1001
c
      do 100 i = 2, n
  100 e(i-1) = e(i)
c
      e(n) = 0.0d0
c
      do 290 l = 1, n
         j = 0
c ..... look for small sub-diagonal element .....
  105 do 110 m = l, n
            if (m .eq. n) go to 120
            tst1 = dabs(d(m)) + dabs(d(m+1))
            tst2 = tst1 + dabs(e(m))
            if (tst2 .eq. tst1) go to 120
  110 continue
c
  120 p = d(l)
```

```
337         if (m .eq. l) go to 215
338         if (j .eq. 30) go to 1000
339         j = j + 1
340 c ..... form shift .....
341         g = (d(l+1) - p) / (2.0d0 * e(l))
342         r = pythag(g,1.0d0)
343         g = d(m) - p + e(l) / (g + dsign(r,g))
344         s = 1.0d0
345         c = 1.0d0
346         p = 0.0d0
347         mml = m - l
348 c ..... for i=m-1 step -1 until l do -- .....
349         do 200 ii = 1, mml
350            i = m - ii
351            f = s * e(i)
352            b = c * e(i)
353            r = pythag(f,g)
354            e(i+1) = r
355            if (r .eq. 0.0d0) go to 210
356            s = f / r
357            c = g / r
358            g = d(i+1) - p
359            r = (d(i) - g) * s + 2.0d0 * c * b
360            p = s * r
361            d(i+1) = g + p
362            g = c * r - b
363   200 continue
364 c
365         d(l) = d(l) - p
366         e(l) = g
367         e(m) = 0.0d0
368         go to 105
369 c ..... recover from underflow .....
370   210 d(i+1) = d(i+1) - p
371         e(m) = 0.0d0
372         go to 105
373 c ..... order eigenvalues .....
374   215 if (l .eq. 1) go to 250
375 c ..... for i=l step -1 until 2 do -- .....
376         do 230 ii = 2, l
377            i = l + 2 - ii
378            if (p .ge. d(i-1)) go to 270
379            d(i) = d(i-1)
380   230 continue
381 c
382   250 i = 1
383   270 d(i) = p
384   290 continue
385 c
386       go to 1001
387 c ..... set error -- no convergence to an
388 c eigenvalue after 30 iterations .....
389  1000 ierr = l
390  1001 return
391       end
392
393 c-------------------------------------------------------------------------------
394
395       double precision function pythag(a,b)
396       double precision a,b
397 c
398 c finds dsqrt(a**2+b**2) without overflow or destructive underflow
399 c
400       double precision p,r,s,t,u
401       p = dmax1(dabs(a),dabs(b))
402       if (p .eq. 0.0d0) go to 20
403       r = (dmin1(dabs(a),dabs(b))/p)**2
404    10 continue
```

```
405         t = 4.0d0 + r
406         if (t .eq. 4.0d0) go to 20
407         s = r/t
408         u = 1.0d0 + 2.0d0*s
409         p = u*p
410         r = (s/u)**2 * r
411       go to 10
412    20 pythag = p
413       return
414       end
```