PERIMETER SCHOLARS INTERNATIONAL
April 5-23, 2010, Course on "Quantum Spin Simulations"

Anders W. Sandvik, Boston University

PART2: Exact diagonalization studies

Studies of small spin chains

Using basis states incorporating conservation laws (symmetries)
- magnetization conservation, momentum states, parity, spin inversion
- discussion without group theory
  - only basic quantum mechanics and common sense needed

Key elements of an actual program implementation
- pseudocode segments; actual programs available for practice

How to characterize different kinds of ground states
- critical ground state of the Heisenberg chain
- quantum phase transition to a valence-bond solid in a J_1-J_2 chain

Complete diagonalization of the hamiltonian

To find the ground state (maybe excitations, T>0 properties)
of the Heisenberg S=1/2 chain

$$H = J\sum_{i=1}^{N} \mathbf{S}_i \cdot \mathbf{S}_{i+1} = J\sum_{i=1}^{N} [S_i^x S_{i+1}^x + S_i^y S_{i+1}^y + S_i^z S_{i+1}^z],$$

$$= J\sum_{i=1}^{N} [S_i^z S_{i+1}^z + \tfrac{1}{2}(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+)]$$

Simplest way; enumerate the states

- construct the hamiltonian matrix using bits

$$|0\rangle = |\downarrow,\downarrow,\downarrow,\ldots,\downarrow\rangle \quad (=0\ldots000)$$
$$|1\rangle = |\uparrow,\downarrow,\downarrow,\ldots,\downarrow\rangle \quad (=0\ldots001)$$
$$|2\rangle = |\downarrow,\uparrow,\downarrow,\ldots,\downarrow\rangle \quad (=0\ldots010)$$
$$|3\rangle = |\uparrow,\uparrow,\downarrow,\ldots,\downarrow\rangle \quad (=0\ldots011)$$

$$H_{ij} = \langle i|H|j\rangle$$
$$i,j = 0,\ldots,2^N-1$$

bit representation perfect for S=1/2 systems

- use >1 bit/spin for S>1/2, or integer vector
- construct H by examining/flipping bits

Diagonalizing the matrix

- on the computer
- gives the eigenvalues and eigenvectors

If U is the matrix whose columns are the eigenvectors of H, then

$$\langle n|A|n\rangle = [U^{T*}AU]_{nn}$$

is the expectation value of some operator A in the n:th eigenstate

Use some "canned" diagonalization subroutine

- Useful subroutines available at http://gams.nist.gov
- Matrix U and eigenvalues delivered

Problem: Matrix size $M=2^N$ becomes too large quickly

- maximum N≈20
- M^2 matrix elements to store, time to diagonalize $\propto M^3$

Pseudocode; construction of the hamiltonian matrix

Let **a[i]** refer to the **i:th** bit of an **integer a**

Define a function **flip**(a,i,j)

- "flips" (0↔1) bits i and j of the integer a
- In F90 the bit-level function **ieor(a,2**i)** can be used to flip bit i of a
- bits i and j can be flipped using **ieor(a,2**i+2**j)**

				j			i	
a	0	1	0	1	0	0	1	1
$2^i + 2^j$	0	0	0	1	1	0	0	0
$\mathrm{ieor}(a, 2^i + 2^j)$	0	1	0	0	1	0	1	1

The S=1/2 Heisenberg chain hamiltonian can be constructed according to:

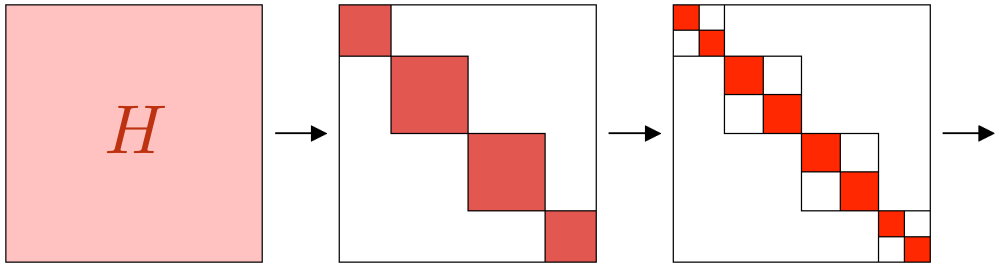
```
do a = 0, 2^N − 1
    do i = 0, N − 1
        j = mod(i + 1, N)
        if (a[i] = a[j]) then
            H(a, a) = H(a, a) + 1/4
        else
            H(a, a) = H(a, a) − 1/4
            b = flip(a, i, j); H(a, b) = 1/2
        endif
    enddo
enddo
```

j is the “right” nearest-neighbor of i

- periodic boundary conditions

Using conservation laws (symmetries) for block-diagonalization

We can choose the basis in such a way that the H becomes block-diagonal



- the blocks can be diagonalized individually
- we can reach larger N (but not much larger, N≈40 is max)

Simplest example; magnetization conservation:

$$m_z = \sum_{i=1}^{N} S_i^z$$

- blocks correspond to fixed values of m_z
- no H matrix elements between states of different m_z
- block-diagonalization just amounts to including states with only given m_z

Number of states in the largest block (m_z =0):

$$\frac{N!}{(N/2)!(N/2)!}$$

Other symmetries (conserved quantum numbers)

- can be used to further split the blocks
- but more complicated
  - basis states have to be constructed to obey symmetries

Pseudocode: using magnetization conservation

Constructing the basis in the block of $n_\uparrow$ spins ↑

Store state-integers in ordered list **s_a, a=1,....,M**

```
do s = 0, 2^N − 1
    if (∑_i s[i] = n_↑) then a = a + 1; s_a = s endif
enddo
M = a
```

Example; N=4, $n_\uparrow$=2

s_1=3 (0011)
s_2=5 (0101)
s_3=6 (0110)
s_4=9 (1001)
s_5=10 (1010)
s_6=12 (1100)

How to locate a state (given integer s) in the list?

- stored map s→a would be too big for s=0,...,2^N-1
- instead, we search the list s_a (here simplest way)

```
subroutine findstate(s, b)
b_min = 1; b_max = M
do
    b = b_min + (b_max − b_min)/2
    if (s < s_b) then
        b_max = b − 1
    elseif (s > s_b) then
        b_min = b + 1
    else
        exit
    endif
enddo
```

Finding the location ***b***
of a state-integer ***s*** in the list

- using bisection in the ordered list

Pseudocode; hamiltonian construction

- recall: states labeled a=1,...,M
- corresponding state-integers (bit representation) stored as s_a
- bit i, $s_a[i]$, corresponds to S^z_i

```
do a = 1, M
    do i = 0, N − 1
        j = mod(i + 1, N)
        if (s_a[i] = s_a[j]) then
            H(a, a) = H(a, a) + 1/4
        else
            H(a, a) = H(a, a) − 1/4
            s = flip(s_a, i, j)
            call findstate(s, b)
            H(a, b) = H(a, b) + 1/2
        endif
    enddo
enddo
```

loop over states

loop over sites

check bits of state-integers

state with bits i and j flipped