Matrix calculation

The Heisenberg model

Matrix calculation

The Heisenberg model

- Interactions in a long chain of N quantistic spins
- There are real systems that can be studied by this simple model
- The quantum state of the chain can be expressed in terms (a product) of the single spin states, eigenvectors of σ^z, with eigenvalues ± 1. Each spin points up or down.
- E.g., typical configuration for 9 spins: $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$
- There are 2^N configurations (states for the whole chain)

The hamiltonian of the Heisenberg model is

$$\mathcal{H} = 2\sum_{j=1}^{N} \vec{S}_j \cdot \vec{S}_{j+1} = \frac{1}{2}\sum_{j=1}^{N} \vec{\sigma}_j \cdot \vec{\sigma}_{j+1}$$

where $\vec{S}_j = \frac{1}{2}\vec{\sigma}_j$ is the spin operator acting on spin j.

- The energy of a chain depends on whether adjacent spins are parallel or antiparallel
- A pair like $\uparrow\downarrow$ or $\downarrow\uparrow$ contributes $-\frac{1}{2}$. Pairs like $\uparrow\uparrow$ or $\downarrow\downarrow$ contribute $+\frac{1}{2}$

$\vec{\sigma} \equiv (\sigma^x, \sigma^y, \sigma^z) \equiv (\sigma^1, \sigma^2, \sigma^3)$ are the 2×2 *Pauli spin matrices*.

It can be shown that $\vec{\sigma}_a \cdot \vec{\sigma}_b = 2(\sigma_a^+\sigma_b^- + \sigma_a^-\sigma_b^+) + \sigma_a^z\sigma_b^z$,
where $\sigma_l^{\pm} \equiv \frac{1}{2}(\sigma_l^x \pm i\sigma_l^y)$ flip the l-th spin up (or down).

Notice that the spin states as we defined are a complete set of states (a basis) *but are not* eigenstates of $\mathcal{H}$. So $\mathcal{H}$ is not diagonal on this basis.

$\mathcal{H}$ is can be represented as a $2^N \times 2^N$ matrix H

The energy levels of the quantum system are the eigenvalues of H.

Given two spin configurations of the string (C_l and C_m), the (l,m) element of H ($\mathrm{H}_{l,m}$) is the matrix element $\langle C_l \mid \mathcal{H} \mid C_m \rangle$

Let's try for $N = 2$...

The general *practical* **rule to build** H **is:**

- Each row and column of H is labelled by a pattern of spins (spins state) and an element of H can be seen as "linking" 2 patterns.
- The diagonal elements of the matrix are computed as $+\frac{1}{2}$ for each $\uparrow\uparrow$ or $\downarrow\downarrow$, and $-\frac{1}{2}$ for every $\uparrow\downarrow$ or $\downarrow\uparrow$.
- The only non-zero entries off the diagonal correspond to patterns that can be generated by changing $\uparrow\downarrow$ into $\downarrow\uparrow$ or $\downarrow\uparrow$ into $\uparrow\downarrow$. In this case, the off-diagonal entry is then 1.

Example: configuration $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$

$\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ $\quad$ $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$

$\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ $\quad$ $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ $\quad$ $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$

$\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ $\quad$ $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$ $\quad$ $\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\downarrow\uparrow$

Entry $2 \times \frac{1}{2} + 6 \times \left(-\frac{1}{2}\right) = -2$ on the diagonal of the matrix

Example: configuration ↑↑↓↑↓↓↑↓↑

↑↑↓↑↓↓↑↓↑   ↑↑↓↑↓↓↑↓↑

↑↑↓↑↓↓↑↓↑   ↑↑↓↑↓↓↑↓↑   ↑↑↓↑↓↓↑↓↑

↑↑↓↑↓↓↑↓↑   ↑↑↓↑↓↓↑↓↑   ↑↑↓↑↓↓↑↓↑

Entry $2 \times \frac{1}{2} + 6 \times \left(-\frac{1}{2}\right) = -2$ on the diagonal of the matrix

Entries equal to 1 for configurations

↑↓↑↑↓↓↑↓↑   ↑↑↑↓↓↓↑↓↑   ↑↑↓↓↑↓↑↓↑

↑↑↓↑↓↑↓↓↑   ↑↑↓↑↓↓↓↑↑   ↑↑↓↑↓↓↑↑↓

For example, for $N = 4$ the dimension of the matrix is $2^N \times 2^N = 16 \times 16$ ($= 256$ elements!).

The matrix form of the Hamiltonian of the quantum Heisenberg model of spins $\vec{S}_j = (S_j^x, S_j^y, S_j^z)$ on a one-dimensional lattice of $N = 4$ sites with periodic boundary conditions $\vec{S}_{N+1} = \vec{S}_1$ is given by $\Rightarrow$

$$
\begin{array}{cccccccccccccccc|c}
\downarrow\downarrow\downarrow\downarrow & \downarrow\downarrow\downarrow\uparrow & \downarrow\downarrow\uparrow\downarrow & \downarrow\downarrow\uparrow\uparrow & \downarrow\uparrow\downarrow\downarrow & \downarrow\uparrow\downarrow\uparrow & \downarrow\uparrow\uparrow\downarrow & \downarrow\uparrow\uparrow\uparrow & \uparrow\downarrow\downarrow\downarrow & \uparrow\downarrow\downarrow\uparrow & \uparrow\downarrow\uparrow\downarrow & \uparrow\downarrow\uparrow\uparrow & \uparrow\uparrow\downarrow\downarrow & \uparrow\uparrow\downarrow\uparrow & \uparrow\uparrow\uparrow\downarrow & \uparrow\uparrow\uparrow\uparrow & \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \downarrow\downarrow\downarrow\downarrow \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \downarrow\downarrow\downarrow\uparrow \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \downarrow\downarrow\uparrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \downarrow\downarrow\uparrow\uparrow \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \downarrow\uparrow\downarrow\downarrow \\
0 & 0 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \downarrow\uparrow\downarrow\uparrow \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \downarrow\uparrow\uparrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \downarrow\uparrow\uparrow\uparrow \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \uparrow\downarrow\downarrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \uparrow\downarrow\downarrow\uparrow \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & -2 & 0 & 1 & 0 & 0 & 0 & \uparrow\downarrow\uparrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \uparrow\downarrow\uparrow\uparrow \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \uparrow\uparrow\downarrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \uparrow\uparrow\downarrow\uparrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \uparrow\uparrow\uparrow\downarrow \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & \uparrow\uparrow\uparrow\uparrow
\end{array}
$$

Matrix size $2^N \times 2^N$. Can easily get $N = 16$

Physical systems have large values of N so we need $N \to \infty$.

Matrix size $2^N \times 2^N$.   Can easily get $N = 16$

Physical systems have large values of N so we need $N \to \infty$.

For one-dimensional chains we can get N large enough to extrapolate to $N = \infty$

For two or three-dimensional lattices of spins and the size of the lattice is then $N^{1/2} \times N^{1/2}$ or $N^{1/3} \times N^{1/3} \times N^{1/3}$

It is impossible to get N large enough to approximate real systems.

Matrix size $2^N \times 2^N$. Can easily get $N = 16$

Physical systems have large values of N so we need $N \to \infty$.

For one-dimensional chains we can get N large enough to extrapolate to $N = \infty$

For two or three-dimensional lattices of spins and the size of the lattice is then $N^{1/2} \times N^{1/2}$ or $N^{1/3} \times N^{1/3} \times N^{1/3}$

It is impossible to get N large enough to approximate real systems.

This is a simple problem whose accurate solution is beyond current computers.

Using symmetry and hard work will get $N \sim 24$

For larger matrices Monte Carlo is the only possibility.

Computing Eigenvalues of Matrices

Many physics problems require the calculation of the eigenvalues of a matrix.

Computing Eigenvalues of Matrices

Many physics problems require the calculation of the eigenvalues of a matrix.

Find N vectors $\mathbf{u}_n$ and numbers λ_n associated with the $N \times N$ matrix A such that

$$\mathsf{A}\mathbf{u}_n = \lambda_n \mathbf{u}_n .$$

The general methods are beyond this course and you would normally use a library program which converts the matrix into a tridiagonal one with the same eigenvalues.

A tridiagonal matrix has all elements zero except on the diagonal and one stripe on either side.

This step requires all the matrix in memory at once, so limits the maximum size of matrix.

One method for BIG matrices

Suppose $\mathbf{x}^{(0)}$ is a random unit vector

Sequence of vectors, also of unit length, $\mathbf{x}^{(k)}$ calculated by

$$\mathbf{x}^{(k+1)} = \frac{1}{L_k}\mathsf{A}\mathbf{x}^{(k)},$$

L_k is the scaling constant needed to make $\mathbf{x}^{(k+1)}$ a unit vector.

If the sequence converges to a fixed vector so that $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ then $\mathbf{x}^{(k)}$ is one of the eigenvectors and L_k converges to the corresponding eigenvalue.

The sequence does converge to the vector with the largest value of $|\lambda|$.

This algorithm involves operating with the matrix A on a vector.

Only the vectors $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ need be in memory (A can be on disk).

In the important applications most of the elements of the matrix are zero and do not appear in the calculation.

Multiplication by A need not be evaluated as a matrix operation provided we have an algorithm for getting A$\mathbf{x}$ from $\mathbf{x}$.

The limit on the matrix size is the memory to hold the two vectors.

Beyond this Monte Carlo methods are possible!

Computer Exercise: The Heisenberg model

In the computer we use 0 and 1 to represent $\downarrow$ and $\uparrow$; a spin configuration is simply an binary integer.

The matrix H is very sparse we do not store it but use a procedure takes in a vector $\mathbf{v}$ and returns the vector $\mathbf{u}$ equal to H$\mathbf{v}$.

1. Diagonalisation with the whole matrix in memory

   Possible for very small N.

2. Iterative method for the ground state

   Compute λ/N. You will run out of memory somewhere before $N = 20$.

3. Extrapolate to $N = \infty$

   Record the ground state energy for various N, including both odd and even values, and extrapolate to $N = \infty$.

4. First excited state

- Eigenvectors are orthogonal.
- Find the ground state eigenvector $\mathbf{u_0}$.
- Repeat the iteration including an extra step to force $\mathbf{v}$ to be orthogonal to $\mathbf{u_0}$.
- If $\mathbf{u_0}$ is a unit vector then $\mathbf{v} - (\mathbf{u_0} \cdot \mathbf{v})\mathbf{u_0}$ is orthogonal to $\mathbf{u_0}$. (A piece of code is included in the program to do this.)

Extrapolate as before.

Overview of program (megaeig.py)

1. Create bit patterns corresponding to adjacent pairs of spins
2. Create whole $2^N \times 2^N$ matrix in memory and get energies
3. Iterative method for ground state
4. Space to fill in extension for first excited state
5. Code to carry out operation $\mathbf{u} = \mathrm{H} \times \mathbf{v}$
6. Code to normalise a vector
7. Code to make $\mathbf{v}$ orthogonal to $\mathbf{u_0}$ (by removing its component "parallel" to $\mathbf{u_0}$).

Operate with matrix on vector

```
def matrix(u, v):
    "Computes u=Hv where H is hamiltonian"
    for i in range(0,M):
        sum=0.0
        for j in range(0,N):
            p=pair[j]
            m=i & p   # & is bitwise AND
            if not ( (m==0) or (m==p) ):
                sum=sum + v[i^p] - 0.5*v[i]
                # ^ is bitwise XOR
            else:
                sum=sum + 0.5*v[i]

        u[i]=sum
```

```
# Main Program
N=10
M=2**N
pair=array([3<<i for i in range(0,20)], int) ; pair[N-1]=(M/2)+1
## << left shift operator: add a trailing 0 in binary rapresentation
# Compute eigenvalues using full matrix in memory
H=zeros((M,M), float)
u=zeros(M, float)
for j in range(0,M):                     # Construct full matrix
v=zeros(M, float)
  v[j]=1.0
  matrix(u, v)
  H[:,j]=u[:]
(H, val)=mateig(H)                       #compute eigenvalues
print val/N
```

```
# Iterative method for ground state using 2 vectors only in memory
u=zeros(M, float)
v=array([uniform(1) for i in range(0,M)], float)
normalise(v)

E=1e10
lastE=0
while abs(E-lastE) > 1e-5*abs(E):
    lastE=E
    matrix(u, v)
    E=normalise(u)
    v[:]=u[:]          # copy array u to v
print E/N
```

Main Points

- How finite difference approximate first and second derivatives.
- The merits of finite difference solutions compared with theoretical ones.
- Lots of physics problems translate into linear equations.
- Some techniques for handling linear equations on computers.
- Simple physics problems can easily be beyond current average computers.
- Finding eigenvalues of very big matrices.
- Extrapolation to infinite systems from finite ones.