ICTS INTERNATIONAL CENTRE for THEORETICAL SCIENCES
TATA INSTITUTE OF FUNDAMENTAL RESEARCH

Computational Methods for Quantum Many-Body Physics

Sthitadhi Roy[1] and Arnab Sen[2]

May 24, 2023

[1]ICTS-TIFR, Bengaluru
[2]IACS, Kolkata

Lecture 2

Recap of Lecture 1

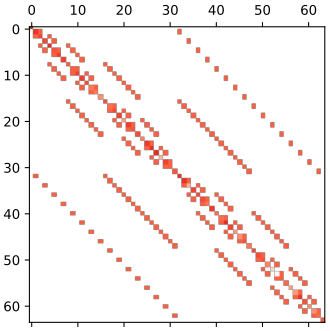
- overview of physics and kinds of physical systems we will be treating
- overview of techniques we will be discussing
- **Exact diagonalisation** for spin systems
  - Constructing the basis states
  - Encoding them as binary strings
  - Efficient ways of tabulating them so that they are easy to look up [Lin Tables]
  - Implementing $U(1)$ symmetry
  - constructing the Hamiltonian

This lecture: diagonalising the Hamiltonian, and extracting eigenvalues and eigenvectors

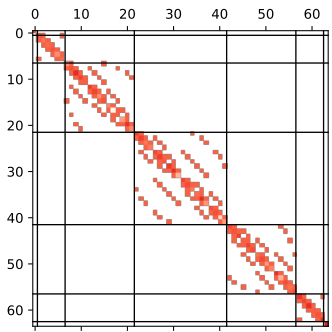
Structure of the Hamiltonian matrix

$$H = J \sum_{i=0}^{L-1} [\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y + \sigma_i^z \sigma_{i+1}^z] + \sum_i h_i \sigma_i^z$$

Plot of Hamiltonian matrix for $L = 6$



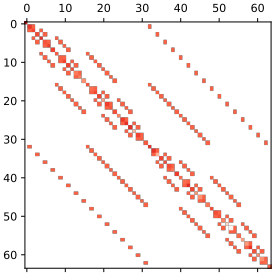
Block diagonalised into total S^z sectors



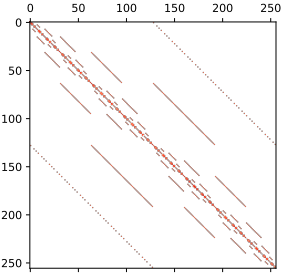
- the Hamiltonian is very sparse
- lots of zeros in the Hamiltonian

Sparsity of the Hamiltonian matrix

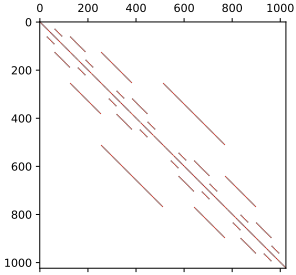
Plot of Hamiltonian matrix for $L = 6$



Plot of Hamiltonian matrix for $L = 8$



Plot of Hamiltonian matrix for $L = 10$

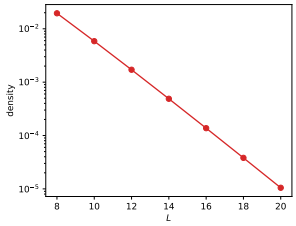


- with increasing system size, the Hamiltonian seems to become more and more sparse
- define

$$\text{density} = \frac{\#\text{ non-zero elements}}{(\text{Hilbert-space dimension})^2}$$

Sparsity of the Hamiltonian matrix

Plot of sparsity vs L



- the density decreases exponentially with L

Why is the Hamiltonian so sparse?

- due to the locality of the Hamiltonian
- from any basis state, we have $\mathcal{O}(L)$ rearrangements possible
- each row has $\mathcal{O}(L)$ non-zero elements
- $$\text{density} \sim \frac{\mathcal{D} \times L}{\mathcal{D}^2} \sim L e^{-L}$$
- even for long-ranged interacting, but local systems

$$H = \sum_{i,j} \frac{J}{|i-j|^\alpha} [\sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y + \sigma_i^z \sigma_{i+1}^z] + \sum_i h_i \sigma_i^z$$

- each row has $\mathcal{O}(L^2)$ non-zero elements

$$\text{density} \sim L^2 e^{-L}$$

Generally Hamiltonians of **locally interacting** quantum systems are sparse $\Rightarrow$ ought to take advantage of that

Sparse matrices are efficient to store and manipulate

Why use sparse matrices?

- storing all the zeros is redundant, instead only store the non-zero elements and their locations (row and column indices)
- memory efficient; fraction of memory needed is only $3 \times$ density
- extremely efficient matrix-vector multiplication (particularly when the sparse matrix is stored in CSR format) $\Rightarrow$ useful because much of quantum mechanics is applying operators (matrices) on states (vectors)

Representing sparse matrices on a computer

COO: Coordinate List

- store 3 lists/arrays of length = number of non-zero elements
  - values of the non-zero element
  - row indices
  - column indices

CSR: Compressed Sparse Row

- again 3 lists/arrays
  - values of the non-zero element
  - column indices
  - row-pointer indices

COO Sparse Matrix: Example

Example matrix

$$\begin{pmatrix} 0 & 0.04 & 0 & 0 & 0 & 0.18 \\ 0.08 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.14 & 1.88 & 0.44 & 0 & 0.62 \\ 0 & 0 & 0 & 1.21 & 0 & 0 \\ 0 & 0 & 0.63 & 0 & 0 & 0 \\ 0 & 1.43 & 0.45 & 0 & 0 & 0.34) \end{pmatrix}$$

COO representation

data	0.04	0.18	0.08	0.14	1.88	0.44	0.62	1.21	0.63	1.43	0.45	0.34
row	0	0	1	2	2	2	2	3	4	5	5	5
col	1	5	0	1	2	3	5	3	2	1	2	5

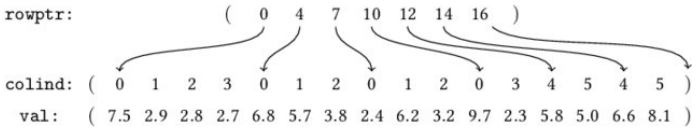
CSR Sparse Matrix

- the structure is slightly different from COO
- the lists containing the matrix elements and the column indices are the same as COO
- the information of the row indices are stored somewhat differently: call it `rowptr`

- length of `rowptr` $= 1+\#$ rows in the matrix
- `rowptr`$= 0$
- `rowptr[i]` $=$ the total number of non-zero elements until row `i`

Example:

$$A = \begin{pmatrix} 7.5 & 2.9 & 2.8 & 2.7 & 0 & 0 \\ 6.8 & 5.7 & 3.8 & 0 & 0 & 0 \\ 2.4 & 6.2 & 3.2 & 0 & 0 & 0 \\ 9.7 & 0 & 0 & 2.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.8 & 5.0 \\ 0 & 0 & 0 & 0 & 6.6 & 8.1 \end{pmatrix}$$



Sparse matrix-vector multiplication: COO

$$Ax = y$$

- recall the three lists, `data, rowind, colind`
- define `nnz` = number of non-zero elements in the sparse matrix
- `data[i]`$= A_{\texttt{rowind[i]},\texttt{colind[i]}}$

pseudo-code for a COO matrix-vector multiplication

```
for(i=0; i<nnz;  i++)
   y[rowind[i]] += data[i]*x[colind[i]]
```

- the number of operations required is `nnz`; recall that $\texttt{nnz}/\mathcal{D}^2$ is extremely small
- already we are getting rid of lots of redundancies

Sparse matrix-vector multiplication: CSR

- recall the three lists again
  - `data`: length is `nnz`
  - `colind`: length is `nnz`
  - `rowptr`: length is `N+1`; `N` is the number of rows in the matrix

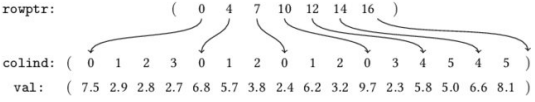
pseudo-code for a CSR matrix-vector multiplication

```
for (i=0; i<N; i++){
   y[i] = 0.0;
   for (j=rowptr[i]; j<rowptr[i+1]; j++)
      y[i] += data[j]*x[colind[j]];
}
```

- we do not have to look up each row of the sparse matrix over and over and again
- already we are getting rid of lots of redundancies

Example

$$A = \begin{pmatrix} 7.5 & 2.9 & 2.8 & 2.7 & 0 & 0 \\ 6.8 & 5.7 & 3.8 & 0 & 0 & 0 \\ 2.4 & 6.2 & 3.2 & 0 & 0 & 0 \\ 9.7 & 0 & 0 & 2.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.8 & 5.0 \\ 0 & 0 & 0 & 0 & 6.6 & 8.1 \end{pmatrix}$$



Brief summary so far...

- Hamiltonians of locally interacting systems have lots of zeros
- Store them as sparse matrices; only keep track of the non-zero elements
- extremely efficient because fraction of filled in elements goes down exponentially with system size
- different formats for storing sparse matrices on a computer: COO, CSR (also CSC, DIA etc.)
- efficient sparse matrix-vector multiplication

How to use the efficient sparse matrix-vector multiplication to extract eigenvalues and eigenvectors

Brief summary so far...

- Hamiltonians of locally interacting systems have lots of zeros
- Store them as sparse matrices; only keep track of the non-zero elements
- extremely efficient because fraction of filled in elements goes down exponentially with system size
- different formats for storing sparse matrices on a computer: COO, CSR (also CSC, DIA etc.)
- efficient sparse matrix-vector multiplication

Lanczos algorithm for Exact Diagonalisation

Basic idea: iteratively converge to the target eigenvalues and eigenvectors

- Power-law iteration
- Basic Lanczos algorithm
- Lanczos algorithm on a truncated Krylov space

An essential point: these iterative procedures converge best to extremal eigenvalues

- ground state physics and low-lying excitations (often of extreme interest) naturally conducive
- for states in the middle of many-body spectrum (relevant for dynamics), we have to transform the hamiltonian (next lecture)
  - Shift-invert ED
  - Polynomially-filtered ED

Power-iteration method

- Consider a Hamiltonian H with eigenvalues and eigenvectors $H|\psi_i\rangle = E_i|\psi_i\rangle$
- Let's say $|\psi_0\rangle$ is the ground state and the Hamiltonian as been scaled/shifted such that E_0 has the maximum magnitude
- Start with a random state $|\phi_0\rangle$ $\Rightarrow$ very unlikely that it will be orthogonal to the ground state.

$$|\phi_0\rangle = \sum_i c_i |\psi_i\rangle\,; \qquad \langle\psi_0|\phi_0\rangle = c_0 \neq 0$$

- Apply the Hamiltonian n times and renormalise the state

$$|\phi_0^{(n)}\rangle = \frac{H^n|\phi_0\rangle}{||H^n|\phi_0\rangle||} = \frac{\sum_i \frac{c_i}{c_0}\left(\frac{E_i}{E_0}\right)^n |\psi_i\rangle}{\left[\sum_i \frac{|c_i|^2}{|c_0|^2}\left(\frac{E_i}{E_0}\right)^{2n}\right]^{1/2}}$$

- For large $n \gg 1$

$$|\phi_0^{(n)}\rangle = |\psi_0\rangle + \text{error}; \qquad \text{error} \sim \max_i \left|\frac{E_i}{E_0}\right|^n$$

- Generate another random state $|\phi_1\rangle$ and orthogonalise to $|\psi_0\rangle$ and repeat to get the next excited state
- works best when the extremal values are well separated: low density of states

Lanczos algorithm for Exact Diagonalisation

Can do much better: basic idea is that the ground state and low-lying excited states live in a small subspace of the entire Hilbert-space

Basic Lanczos algorithm

- Consider a random state as before $|\phi_0\rangle$
- define $a_0 = \langle\phi_0|H|\phi_0\rangle$
- Generate $|\tilde{\phi}_1\rangle$ which is orthogonal to $|\phi_0\rangle$ as $|\tilde{\phi}_1\rangle = H|\phi_0\rangle - |\phi_0\rangle\langle\phi_0|H|\phi_0\rangle$
- Normalise the state $|\phi_1\rangle = |\tilde{\phi}_1\rangle / \sqrt{\langle\tilde{\phi}_1|\tilde{\phi}_1\rangle} = |\tilde{\phi}_1\rangle / b_1$; note that $b_1 = \langle\phi_1|\phi_0\rangle$
- define $a_1 = \langle\phi_1|H|\phi_1\rangle$
- allows to define a reduced Hamiltonian

$$H_{\text{span}[|\phi_0\rangle, H|\phi_0\rangle]} = \begin{pmatrix} a_0 & b_1 \\ b_1 & a_1 \end{pmatrix}$$

- diagonalise the reduced Hamiltonian to get the ground state $|\phi_0^{(1)}\rangle = \alpha|\phi_0\rangle + \beta|\phi_1\rangle$
- repeat the same procedure on span$[|\phi_0^{(1)}\rangle, H|\phi_0^{(1)}\rangle]$ and iterate...

Lanczos algorithm for Exact Diagonalisation

Basic Lanczos algorithm

- minimise on $\text{span}[|\phi_0\rangle, H|\phi_0\rangle]$ to obtain $|\phi_0^{(1)}\rangle$
- minimise on $\text{span}[|\phi_0^{(1)}\rangle, H|\phi_0^{(1)}\rangle]$ to obtain $|\phi_0^{(2)}\rangle \in \text{span}[|\phi_0\rangle, H|\phi_0\rangle, H^2|\phi_0\rangle]$
- minimise on $\text{span}[|\phi_0^{(2)}\rangle, H|\phi_0^{(2)}\rangle]$ to obtain $|\phi_0^{(3)}\rangle \in \text{span}[|\phi_0\rangle, H|\phi_0\rangle, H^2|\phi_0\rangle, H^3|\phi_0\rangle]$
- $\ddots$

A better Lanczos algorithm

- Instead of the iterative minimisation over the two-dimensional subspaces, minimise directly over the Krylov subspace

$$\mathcal{K}^m(|\phi_0\rangle) = \text{span}[|\phi_0\rangle, H|\phi_0\rangle, H^2|\phi_0\rangle, H^3|\phi_0\rangle, \cdots, H^m|\phi_0\rangle]$$

- many-more degrees of freedom $\Rightarrow$ much better and faster convergence

Lanczos algorithm for Exact Diagonalisation in Krylov subspace

How to construct the orthonormal basis in Krylov subspace?

$$
\begin{aligned}
|\phi_0\rangle &\equiv \text{random vector normalised} \\
b_1 |\phi_0^{(1)}\rangle &= H|\phi_0\rangle - a_0 |\phi_0\rangle \\
b_2 |\phi_0^{(2)}\rangle &= H|\phi_0^{(1)}\rangle - a_1 |\phi_0^{(1)}\rangle - b_1 |\phi_0\rangle \\
b_3 |\phi_0^{(3)}\rangle &= H|\phi_0^{(2)}\rangle - a_2 |\phi_0^{(2)}\rangle - b_2 |\phi_0^{(1)}\rangle \\
&\ddots
\end{aligned}
$$

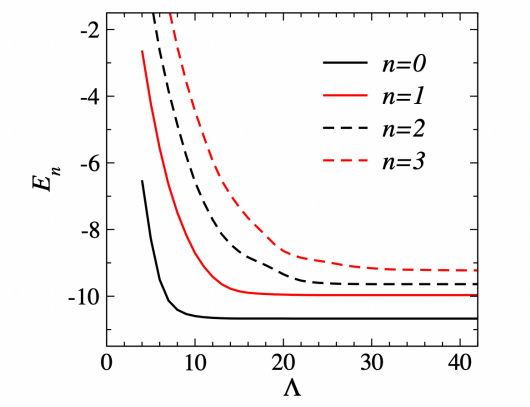
Reduced Hamiltonian in the Krylov space

$$H|\phi_0^{(n)}\rangle = b_n |\phi_0^{(n-1)}\rangle + a_n |\phi_0^{(n)}\rangle + b_{n+1} |\phi_0^{(n+1)}\rangle$$

$$
H_{\mathcal{K}^L(|v_0\rangle)} = \begin{pmatrix}
a_0 & b_1 & 0 & 0 & & 0 & 0 \\
b_1 & a_1 & b_2 & 0 & \cdots & 0 & 0 \\
0 & b_2 & a_2 & b_3 & & 0 & 0 \\
0 & 0 & b_3 & a_3 & & 0 & 0 \\
 & \vdots & & & \ddots & \vdots & \\
0 & 0 & 0 & 0 & & a_{L-1} & b_L \\
0 & 0 & 0 & 0 & \cdots & b_L & a_L
\end{pmatrix}
$$

- tridiagonal matrices are very easy to diagonalise: look up algorithms
- more importantnly, the dimension of the the tridiagonal matrix can be extremely small compared to the original Hilbert-space dimension
- in fact, one can progressively keep increasing the Krylov subspace dimension until all the required eigenvalues/eigenvectors have converged

Lanczos algorithm for Exact Diagonalisation in Krylov subspace



Lanczos convergence for a 24 site Heisenberg chain taking into account some symmetries Sandvik's lecture notes