Potential problem:

The normalization constants N_m can become very large (think of E_0^m)

Solution:

generate the normalized basis directly

- start with $|\phi_0\rangle$ arbitrary, normalized, and then

$$|\phi_1\rangle = \frac{1}{N_1}\big(H|\phi_0\rangle - a_0|\phi_0\rangle\big).$$

$$|\phi_{m+1}\rangle = \frac{1}{N_{m+1}}\big(H|\phi_m\rangle - a_m|\phi_m\rangle - N_m|\phi_{m-1}\rangle\big) = \frac{|\gamma_{m+1}\rangle}{N_{m+1}}$$

The definition of N_m is different, and no b_m:

$$\begin{aligned} a_m &= \langle\phi_m|H|\phi_m\rangle \\ N_m &= \langle\gamma_m|\gamma_m\rangle^{-1/2} \end{aligned}$$

Generate $|\gamma_m\rangle$ first, normalize to get N_{m+1}

The H-matrix is

$$\begin{aligned} \langle\phi_{m-1}|H|\phi_m\rangle &= N_m \\ \langle\phi_m|H|\phi_m\rangle &= a_m \\ \langle\phi_{m+1}|H|\phi_m\rangle &= N_{m+1} \end{aligned}$$

Lanczos basis generation in practice

Here: generate the orthogonal basis **{ϕ_m}** directly

$$|\phi_m\rangle = \sum_{a=1}^{M} \phi_m(a)|a\rangle, \quad m = 0, \ldots, \Lambda$$

in a given symmetry block of size M

The coefficients **$\phi_m(a)$** are stored as Λ+1 vectors of size M

- may store only the vectors **ϕ_{m-1}** and **ϕ_m** to generate **ϕ_{m+1}**
  - but basis has to be re-generated when computing expectation values
- stabilization by "re-orthogonalization" (later) requires storage of all **ϕ_m**

The main computational effort is in acting with the hamiltonian; **$H|\phi_m\rangle$**

- implement as a subroutine **hoperation**(ϕ,γ), where |γ>=H|ϕ>
- state normalization implemented as **normalize**(ϕ,n)
  - ϕ = vector to normalize, n = <ϕ|ϕ> before normalization

Pseudocode; Lanczos basis generation

Initial random state

$$\begin{array}{l}\textbf{do } i = 1, M \\ \quad \phi_0(i) = \textbf{random}[0-1] \\ \textbf{enddo} \\ \textbf{call normalize}(\phi_0, n_0)\end{array}$$

second state

$$\begin{array}{l}\textbf{call hoperation}(\phi_0, \phi_1) \\ a_0 = \langle\phi_0|\phi_1\rangle;\ \phi_1 = \phi_0 - a_0|\phi_1\rangle \\ \textbf{call normalize}(\phi_1, n_1)\end{array}$$

Generate the rest of the states

$$\begin{array}{l}\textbf{do } m = 1, \Lambda - 1 \\ \quad \textbf{call hoperation}(\phi_m, \phi_{m+1}) \\ \quad a_m = \langle\phi_m|\phi_{m+1}\rangle \\ \quad \phi_{m+1} = \phi_{m+1} - a_m\phi_m - n_m\phi_{m-1} \\ \quad \textbf{call normalize}(\phi_{m+1}, n_{m+1}) \\ \textbf{enddo}\end{array}$$

Note: the H-matrix can be constructed and diagonalized after each step

- follow evolution of energy versus Λ
- stop based on some convergence criterion on E_0 (or higher energy)
- expectation values converge slower than energies

The subroutine **hoperation**(ϕ,γ) implements

$$H|\phi\rangle = |\gamma\rangle = \sum_{a=1}^{M}\sum_{b=1}^{M}\phi(a)\langle b|H|a\rangle|b\rangle \qquad |\phi\rangle = \sum_{a=1}^{M}\phi(a)|a\rangle$$

in a given symmetry block (M = block size)

We do not want to store *H* as an *M×M* matrix (too big). Two options:
- carry out the operations on the fly; only the vectors are stored
- store *H* in a compact form; only non-0 elements (sparse matrix)

Storing *H* speeds up the Lanczos iterations
- but may require a lot of memory

Compact storage of H: For each ***a=1,M***
- e_a is the number of non-0 elements $\langle \mathrm{b}|\mathrm{H}|\mathrm{a}\rangle$
- labels $i=s_a+1,s_a+e_a$ will refer to these matrix elements; $\mathrm{s_a} = \sum_{\mathrm{c}=1}^{\mathrm{a}-1}\mathrm{e_a}$
- ***H*(i)** contains the values of the matrix elements $\langle \mathrm{b}|\mathrm{H}|\mathrm{a}\rangle$
- ***B*(i)** contains the corresponding "target" state index ***b***
- The hamiltonian is symmetric
  - store only elements with b ≤ a (divide diagonal elements by 2)

Pseudocode; hamiltonian operation with compact storage

```
subroutine hoperation(φ, γ)
γ = 0; i = 0
do a = 1, M
    do j = 1, e_a
        i = i + 1
        γ(B(i)) = γ(B(i)) + H(i)φ(a)
        γ(a) = γ(a) + H(i)φ(B(i))
    enddo
enddo
```

$$H|\phi\rangle = |\gamma\rangle = \sum_{a=1}^{M}\sum_{b=1}^{M}\phi(a)\langle b|H|a\rangle|b\rangle$$

Further storage compactification possible

- small number of different elements
- use mapping $\langle \mathbf{b}|\mathbf{H}|\mathbf{a}\rangle \rightarrow \text{integer}$
- many operations on Ia> give same Ib>
  - add up all contributions before storing

Operator expectation values

Diagonalizing the tri-diagonal matrix → eigenstates in the Lanczos basis

- eigenvectors v_n, energies E_n
- only some number of low-energy states (<< Λ) are correct eigenstates of H

To compute expectation values we normally go back to the original basis

$$\psi_n(a) = \sum_{m=0}^{\Lambda} v_n(m)\phi_m(a), \qquad a = 1, \ldots, M$$

To compute $\langle \psi_\mathbf{n} | \mathbf{O} | \psi_\mathbf{n} \rangle$ first construct

$$O|\psi_n\rangle = |\psi_n^O\rangle = \sum_{a=1}^{M} \psi_n(a) O|a\rangle$$

$$= \sum_{a=1}^{M}\sum_{b=1}^{M} \psi_n(a)|b\rangle\langle b|O|a\rangle$$

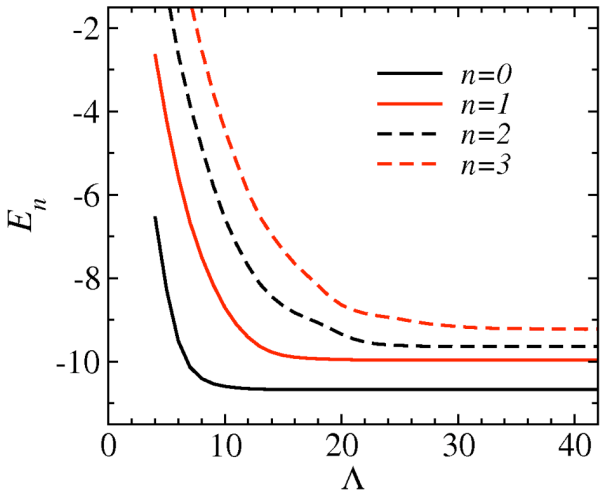
<bIOIa> done exactly as when constructing of the H matrix

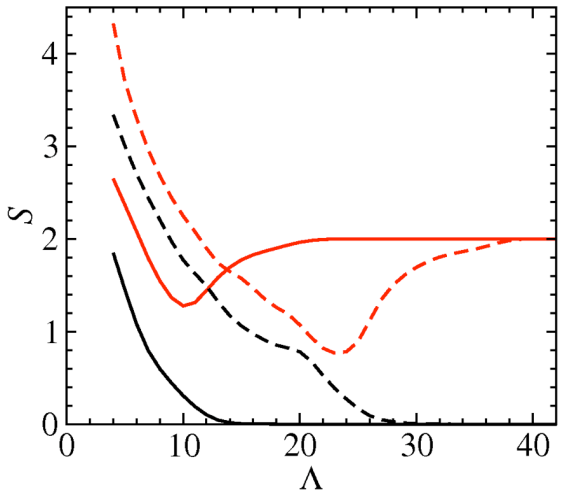
$$= \sum_{b=1}^{M} \psi_n^O(b)|b\rangle \qquad \psi_n^O(b) = \sum_{a=1}^{M} \psi_n(a)\langle b|O|a\rangle$$

Then evaluate the scalar product

$$\langle \psi_n|O|\psi_n\rangle = \langle \psi_n|\psi_n^O\rangle = \sum_{a=1}^{M} \psi_n(a)\psi_n^O(a)$$

Convergence properties of the Lanczos method

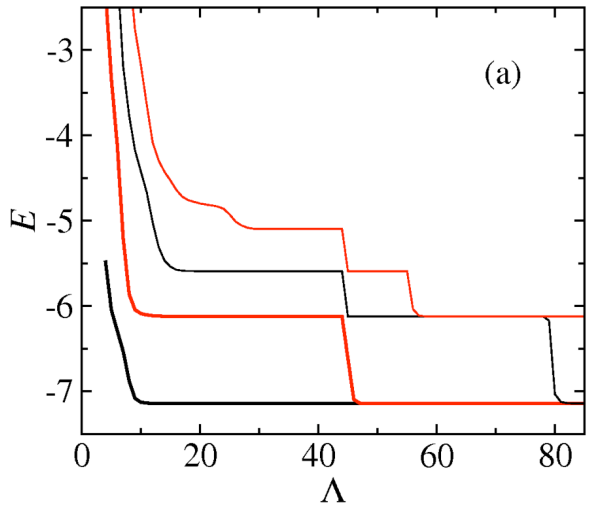


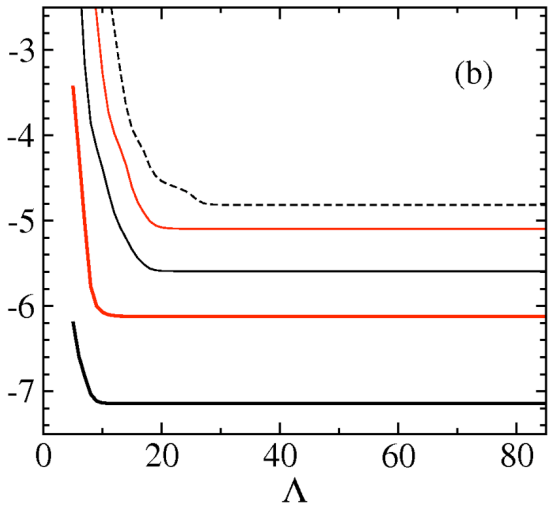


Example; 24-site chain
$m_z = 0$, $k = 0$, $p = 1$, $z = 1$
block size M=28416

Ground state converges first, then successively excited states

Loss of orthogonality: accumulation of numerical error → basis becomes non-orthogonal
- higher states collapse down onto lower ones
- can be cured with re-orthogonalization





Example; 16-site chain
$m_z = 0$, $k = 0$, $p = 1$, $z = 1$
block size M=212
- (a) non-orthogonality
- (b) re-orthogonalized

Re-orthogonalization procedure

For each state generated, remove all components of prior states, *i=1,...,m*

- easy if we work with the normalized basis and all states are stored

$$|\phi_m\rangle \rightarrow \frac{|\phi_m\rangle - q|\phi_i\rangle}{1-q^2}, \qquad q = \langle\phi_i|\phi_m\rangle$$

Pseudocode: modify state generation

```
do m = 1, Λ − 1
    call hoperation(φ_m, φ_{m+1})
    a_m = ⟨φ_m|φ_{m+1}⟩; φ_{m+1} = φ_{m+1} − a_m φ_m − n_m φ_{m−1}
    call normalize(φ_{m+1}, n_{m+1})
    do i = 1, m
        q = ⟨φ_{m+1}|φ_i⟩; φ_{m+1} = (φ_{m+1} − q φ_i)/(1 − q^2)
    enddo
enddo
```

Note: the Lanczos method can only generate a single state of a multiplet

- some random linear combination of degenerate states

Example: 2 degenerate states i, j:

$$H^\Lambda|\Psi\rangle = \sum_{m\neq i,j} c_m E_m^\Lambda |\psi_m\rangle + E_{i,j}^m (c_i|\psi_i\rangle + c_j|\psi_j\rangle)$$

The mixing of the duplet is determined by c_i, c_j of the initial state