Qbasis Manual

Zhentao Wang

September 5, 2017

Contents

Chapter 1

Introduction

Chapter 2

Model representation

2.1 Matrix represenataion of local Hilbert Space

Fermi-Hubbard Model

$$\mathcal{H} = -t \sum_{\langle ij \rangle} \sum_{\sigma} (c^{\dagger}_{i\sigma} c_{j\sigma} + h.c.) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}. \tag{2.1.1}$$

Local Hilbert space is 4-dimensional:

$$\{|0\rangle, |\uparrow\rangle, |\downarrow\rangle, |\uparrow\downarrow\rangle\}. \tag{2.1.2}$$

In this basis:

$$c_{\uparrow} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.1.3a}$$

$$c_{\downarrow} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{2.1.3b}$$

From these two opeators as input, the code is able to automatically derive the following operators:

$$c^{\dagger}_{\uparrow} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.1.4a}$$

$$c^{\dagger}_{\downarrow} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \tag{2.1.4b}$$

$$n_\uparrow = \begin{pmatrix} 0 & & & \\ & 1 & & \\ & & 0 & \\ & & & 1 \end{pmatrix}, \tag{2.1.4c}$$

$$n_\downarrow = \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}. \tag{2.1.4d}$$

t-J Model

$$\mathcal{H} = -t \sum_{\langle ij \rangle} \sum_{\sigma} (c^\dagger_{i\sigma} c_{j\sigma} + h.c.) + J \sum_{\langle ij \rangle} \left[\frac{S^+_i S^-_j + S^-_i S^+_j}{2} + S^z_i S^z_j - \frac{1}{4} n_i n_j \right], \tag{2.1.5}$$

where

$$S^+_i = c^\dagger_{i\uparrow} c_{i\downarrow}, \tag{2.1.6a}$$

$$S^-_i = c^\dagger_{i\downarrow} c_{i\uparrow}, \tag{2.1.6b}$$

$$S^z_i = \frac{1}{2} \left(c^\dagger_{i\uparrow} c_{i\uparrow} - c^\dagger_{i\downarrow} c_{i\downarrow} \right). \tag{2.1.6c}$$

Local Hilbert space is 3-dimensional:

$$\{|0\rangle, |\uparrow\rangle, |\downarrow\rangle\}. \tag{2.1.7}$$

In this basis:

$$c_\uparrow = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.1.8a}$$

$$c_\downarrow = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{2.1.8b}$$

and all other derived operators can be derived automatically by the code.

Spinless Fermion

$$\mathcal{H} = -t \sum_{\langle ij \rangle} (c^\dagger_i c_j + h.c.) + V_1 \sum_{\langle ij \rangle} n_i n_j. \tag{2.1.9}$$

Local Hilbert space is 2-dimensional:

$$\{|0\rangle, |1\rangle\}. \tag{2.1.10}$$

In this basis:

$$c = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{2.1.11}$$

Bose-Hubbard Model

$$\mathcal{H} = -t \sum_{\langle ij \rangle} (b_i^\dagger b_j + h.c.) + \frac{U}{2} \sum_i n_i (n_i - 1) \tag{2.1.12}$$

Local Hilbert space restricted to at most N_{max} bosons:

$$\{|0\rangle, |1\rangle, \ldots, |N_{max}\rangle\}. \tag{2.1.13}$$

In this basis:

$$b = \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \sqrt{N_{max}} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.1.14}$$

and other operators should be automatically derived.

2.2 Lin table

Chapter 3

Finite size cluster

Chapter 4

Symmetry

4.1 Translation symmetry

4.2 Weisse table

Chapter 5

Eigenvalue problem

5.1 Lanczos algorithm

5.1.1 Description

The m-step Lanczos algorithm performs a factorization in the Krylov subspace:

$$HV = VT + b_m \boldsymbol{v}_m \boldsymbol{e}_m^T, \tag{5.1.1}$$

where H is the $N \times N$ matrix of our problem, $V = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1})$ is $N \times m$ matrix formed by the basis in the Krylov subspace, T is the tridiagonal Hessenberg matrix

$$T = \begin{pmatrix} a_0 & b_1 & & \\ b_1 & a_1 & \ddots & \\ & \ddots & \ddots & b_{m-1} \\ & & b_{m-1} & a_{m-1} \end{pmatrix}, \tag{5.1.2}$$

and $\boldsymbol{e}_m^T = (0, 0, \ldots, 0, 1)$ has only one non-zero element.

Note that $\boldsymbol{v}_i$ are orthonormal, i.e.

$$V^\dagger V = 1. \tag{5.1.3}$$

We can diagonalize the Hessenberg matrix T, giving eigen-pair $(\theta_i,\ \boldsymbol{s}_i)$ satisfying

$$T\boldsymbol{s}_i = \boldsymbol{s}_i \theta_i, \tag{5.1.4}$$

With such information, we can form the Ritz pair $(\theta_i,\ \boldsymbol{y}_i)$ which is a good approximation of the eigenvalues and eigenvectors of the original problem H, where

$$\boldsymbol{y}_i \equiv V\boldsymbol{s}_i. \tag{5.1.5}$$

To estimate the error of the eigenvalues, let's first roughly assume $\boldsymbol{y}_i$ is indeed the true eigenvector with eigenvalue E_i, such that $H\boldsymbol{y}_i = E_i\boldsymbol{y}_i$. Then $\|H\boldsymbol{y}_i - \theta_i\boldsymbol{y}_i\| =$

$|E_i - \theta_i|$,which shows that $\|H\boldsymbol{y}_i - \theta_i \boldsymbol{y}_i\|$ is a good estimate of the error of eigenvalue E_i. Now we relax the assumption about y_i, and calculate $\|H\boldsymbol{y}_i - \theta_i \boldsymbol{y}_i\|$ directly:

$$\begin{aligned} \|H\boldsymbol{y}_i - \theta_i \boldsymbol{y}_i\| &= \|HV\boldsymbol{s}_i - V\boldsymbol{s}_i\theta_i\| = \|HV\boldsymbol{s}_i - VT\boldsymbol{s}_i\| \\ &= \|(HV - VT)\,\boldsymbol{s}_i\| = \left\|b_m \boldsymbol{v}_m \boldsymbol{e}_m^T \boldsymbol{s}_i\right\| \\ &= \left|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i\right|, \end{aligned} \tag{5.1.6}$$

which shows that we can use $\left|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i\right|$ as the error estimation for the Lanczos procedure.

5.1.2 Implementation

Algorithm 5.1 Simple Lanczos algorithm to obtain extreme eigenvalue, 3-vector vesion

Require: Space for 3 vectors in RAM: $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$
Require: An array of pointers $\{\tilde{\boldsymbol{v}}_j \to \boldsymbol{v}_{j \bmod 3}\}$
Require: $\boldsymbol{v}_0$ normalized, $\boldsymbol{v}_1 = \boldsymbol{v}_2 = 0$, $b_0 = 0$.
1: **for** $m = 1, 2, \ldots$ **do**
2: $\tilde{\boldsymbol{v}}_m = -b_{m-1}\tilde{\boldsymbol{v}}_{m-2}$ (copy from $\tilde{\boldsymbol{v}}_{m-2}$)
3: $\tilde{\boldsymbol{v}}_m = H\tilde{\boldsymbol{v}}_{m-1} + \tilde{\boldsymbol{v}}_m$
4: $a_{m-1} = (\tilde{\boldsymbol{v}}_{m-1}, \tilde{\boldsymbol{v}}_m)$
5: $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m - a_{m-1}\tilde{\boldsymbol{v}}_{m-1}$
6: $b_m = \|\tilde{\boldsymbol{v}}_m\|$
7: $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m / b_m$
8: calculate $\{\theta_i, s_i\}$
9: for the smallest θ_i, calcualte $|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i|$. Stop if small enough
10: **end for**

Algorithm 5.2 Simple Lanczos algorithm to obtain extreme eigenvalue, 2-vector vesion

Require: Space for 2 vectors in RAM: $\boldsymbol{v}_0, \boldsymbol{v}_1$
Require: An array of pointers $\{\tilde{\boldsymbol{v}}_j \to \boldsymbol{v}_{j \bmod 2}\}$
Require: $\boldsymbol{v}_0$ normalized, $\boldsymbol{v}_1 = 0$, $b_0 = 0$.
1: **for** $m = 1, 2, \ldots$ **do**
2: $\tilde{\boldsymbol{v}}_m = -b_{m-1}\tilde{\boldsymbol{v}}_{m-2}$ (rescale itself, since $\tilde{\boldsymbol{v}}_m$ and $\tilde{\boldsymbol{v}}_{m-2}$ overlap in RAM)
3: $\tilde{\boldsymbol{v}}_m = H\tilde{\boldsymbol{v}}_{m-1} + \tilde{\boldsymbol{v}}_m$
4: $a_{m-1} = (\tilde{\boldsymbol{v}}_{m-1}, \tilde{\boldsymbol{v}}_m)$
5: $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m - a_{m-1}\tilde{\boldsymbol{v}}_{m-1}$
6: $b_m = \|\tilde{\boldsymbol{v}}_m\|$
7: $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m / b_m$
8: calculate $\{\theta_i, s_i\}$
9: for the smallest θ_i, calcualte $|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i|$. Stop if small enough
10: **end for**

In Fig. 5.1.1, we show the evolution of Ritz values and three possible stopping criteria:

1. Relative change of the ground state energy: $|(\theta_0^m - \theta_0^{m-1})/\theta_0^m|$.
2. Relative change of the 1st excited state energy: $|(\theta_1^m - \theta_1^{m-1})/\theta_1^m|$.
3. $|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i|$.

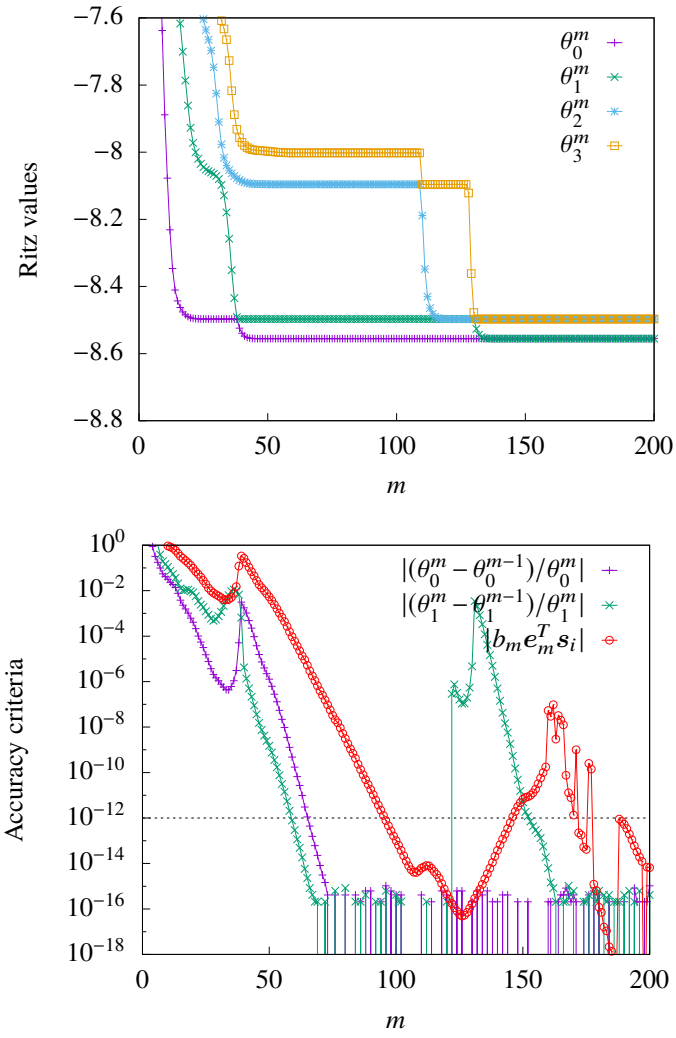

Figure 5.1.1: Evolution of Ritz values for spin-$\frac{1}{2}$ Heisenberg model on 4×4 triangular lattice, with periodic boundary condition and $S_z^{\text{total}} = 0$, $J = 1$. Starting vector is chosen as $(1, 1, 1, \ldots, 1)$.

After convergence, we need extract the ground state eigenvector $\boldsymbol{y}$ (subscript omitted), via

$$\boldsymbol{y} = V\boldsymbol{s} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1})\,\boldsymbol{s}. \tag{5.1.7}$$

Since $\boldsymbol{v}_j$ is not stored along the way, we need one more m-step Lanczos, with the same initial vector $\boldsymbol{v}_0$.

Algorithm 5.3 Simple Lanczos algorithm to obtain eigenvector, $2+1$ vector version

Require: Space for 3 vectors in RAM: $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{y}$
Require: An array of pointers $\{\tilde{\boldsymbol{v}}_j \to \boldsymbol{v}_{j \bmod 2}\}$
Require: $\boldsymbol{v}_0$ same as the one used for eigenvalue, $\boldsymbol{y} = s_0 \boldsymbol{v}_0$, $\boldsymbol{v}_1 = 0$.
for $m = 1, 2, \ldots$ **do**
  $\tilde{\boldsymbol{v}}_m = -b_{m-1} \tilde{\boldsymbol{v}}_{m-2}$
  $\tilde{\boldsymbol{v}}_m = H \tilde{\boldsymbol{v}}_{m-1} + \tilde{\boldsymbol{v}}_m$
  check $a_{m-1} = (\tilde{\boldsymbol{v}}_{m-1}, \tilde{\boldsymbol{v}}_m)$
  $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m - a_{m-1} \tilde{\boldsymbol{v}}_{m-1}$
  check $b_m = \|\tilde{\boldsymbol{v}}_m\|$
  $\tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m / b_m$
  $\boldsymbol{y} = \boldsymbol{y} + s_m \boldsymbol{v}_m$
end for

In exact arithematics, the above procedure should produce the eigenvector, with accuracy given by $|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i|$. Note however, when we run Lanczos again to accumulate $\boldsymbol{y}$, we suffer from finite precision problem, which leads to a much larger error bar in practice $\|H\boldsymbol{y} - E_0 \boldsymbol{y}\| \gg |b_m \boldsymbol{e}_m^T \boldsymbol{s}|$. To overcome this problem, we can choose the initial Lanczos vector to be the approximate ground state ϕ_0, and re-run the above process again. In the refinement cycle, since the initial state ϕ_0 has already a large weight $s_0 \sim 1$, we should expect better accuracy.

In fact, even with a good initial state ϕ_0, it may still take a few refinement cycles (can be as many as 10, each with tens of Lanczos iterations) to have a really good eigenstate. To have faster convergence, we follow the idea of TITPACK and Hphi, using the conjugate gradient method:

Algorithm 5.4 Conjugate gradient method to obtain ground state eigenvector

Require: Space for 4 vectors in RAM: $\boldsymbol{v}_j, \boldsymbol{r}_j, \boldsymbol{p}_j, \tilde{\boldsymbol{p}}_j$
Require: $\boldsymbol{v}_0$ as the initial guess (typically even bad guesses converge fast)
$\boldsymbol{r}_0 = -(H - E_0)\boldsymbol{v}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$, $\gamma_0 = ||\boldsymbol{r}_0||$
for $j = 0, 1, \ldots$ **do**
  $\tilde{\boldsymbol{p}}_j = (H - E_0)\boldsymbol{p}_j$
  $\delta_j = (\boldsymbol{p}_j, \tilde{\boldsymbol{p}}_j)$
  $\alpha_j = \gamma_j^2 / \delta_j$
  $\boldsymbol{v}_{j+1} = \boldsymbol{v}_j + \alpha_j \boldsymbol{p}_j$
  $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j \tilde{\boldsymbol{p}}_j$
  $\beta_j = ||\boldsymbol{r}_{j+1}|| / \gamma_j$
  $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j^2 \boldsymbol{p}_j$
  $\gamma_{j+1} = \beta_j \gamma_j$
end for
Normalize $\boldsymbol{v}_j$. If accuracy not good enough, run once more.

In most cases, the convergence of CG requires almost same steps as Lanczos, even with a random guess of ϕ_0. With this observation, it is no longer necessary to obtain a good initial guess from the 2nd round of Lanczos. In fact, in our code, after obtaining

a good E_0, we will directly use CG with a random initial guess, to get the ground state eigenvector.

We can also obtain the 1st excited state, if we already have a converged ground state $\boldsymbol{\phi}_0$:

Algorithm 5.5 Simple Lanczos algorithm to obtain the eigenvalue of the 1st excited state

Require: Space for 3 vectors in RAM: $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{\phi}_0$
Require: An array of pointers $\{\tilde{\boldsymbol{v}}_j \to \boldsymbol{v}_{j \bmod 2}\}$
Require: $\boldsymbol{\phi}_0$ is the ground state eigenvector, $\boldsymbol{v}_0 \perp \boldsymbol{\phi}_0$, $\boldsymbol{v}_1 = 0$, $b_0 = 0$.
1: **for** $m = 1, 2, \ldots$ **do**
2: $\quad \tilde{\boldsymbol{v}}_m = -b_{m-1}\tilde{\boldsymbol{v}}_{m-2}$
3: $\quad \tilde{\boldsymbol{v}}_m = H\tilde{\boldsymbol{v}}_{m-1} + \tilde{\boldsymbol{v}}_m$
4: $\quad a_{m-1} = (\tilde{\boldsymbol{v}}_{m-1}, \tilde{\boldsymbol{v}}_m)$
5: $\quad \tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m - a_{m-1}\tilde{\boldsymbol{v}}_{m-1}$
6: $\quad b_m = \|\tilde{\boldsymbol{v}}_m\|$
7: $\quad \tilde{\boldsymbol{v}}_m = \tilde{\boldsymbol{v}}_m / b_m$
8: $\quad$ re-orthogonalize $\tilde{\boldsymbol{v}}_m$ against $\boldsymbol{\phi}_0$ if $(\tilde{\boldsymbol{v}}_m, \boldsymbol{\phi}_0)$ no longer small
9: $\quad$ calculate $\{\theta_i, s_i\}$
10: $\quad$ for the smallest θ_i, calcualte $|b_m \boldsymbol{e}_m^T \boldsymbol{s}_i|$. Stop if small enough
11: **end for**

Note: in fact, it takes a lot of Lanczos steps to get a well-converged 1st excited state eigenvector (usually much more than for the ground state). In those cases when we only need the size of the gap $E_1 - E_0$, then we do not have to wait until the convergence of eigenvector.

The eigenvector of the 1st excited state can be obtained as well, as a stright-forward extension of the above algorithms. We leave it as an exercise for the readers.

5.2 Implicitly restarted Arnoldi method

If our goal is only about the ground state, the Lanczos method described in the previous section typically works well. Sometimes, we also need the information of excited states. As we have already discussed, the convergence is already hard even for the 1st excited state for the simple Lanczos method.

The implicitly restarted Arnoldi method (IRAM) is one out of the many approaches to fix this issue. Note that the applications of IRAM go beyond solving the low energy eigenvectors. For details, please see the ARPACK documentary. In this manual, we follow closely the ARPACK implementation of IRAM.

Chapter 6

Code examples

6.1 Exact diagonalization