量子ハイゼンベルグ模型の対角化

1　導入

物理でよく行われると思われる大規模な疎行列の対角化に興味をもったため，そのような問題の一つである 1 次元量子ハイゼンベルグ模型の対角化を今回のレポートのテーマとした（実質的にはテキストの例題 12,13 の解答となっています）．以下では例題 12,13 に沿い，基底状態のエネルギーを通常の行列として扱った場合と疎行列で扱った場合の比較や，サイト数が大きい極限での厳密解と数値解との比較を行った．最後に実装したコードを添付した．

2　ハミルトニアン行列の作成

本レポートで扱う，1 次元 N サイト量子ハイゼンベルグ模型のハミルトニアンは以下の通りである（$\hbar = 1$ とした）．

$$H = \frac{J}{2}\sum_{i=0}^{N-1}(\boldsymbol{S}_i \cdot \boldsymbol{S}_{i+1}) = \frac{J}{2}\sum_{i=0}^{N-1}(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+ + 2S_i^z S_{i+1}^z) \tag{1}$$

このハミルトニアン行列 H の各要素の計算は，スピンの向き $|+\rangle$，$|-\rangle$ をそれぞれ 0,1 に対応させ，ビット演算等を用いながら行った．実装の際には H を式 (1) の最右辺のように捉え，特に非対角成分の非零成分の位置を隣り合うビットを交換させて求めた．H は各サイトのスピンの向きを並べた状態で展開すると $2^N \times 2^N$ 行列となるため，このような方法では全ての要素の作成には $\mathcal{O}(2^{2N})$ かかることになる．

そこで今回はハミルトニアン行列が実対称行列になることに注意し，対角成分の値と上三角成分のうち非零成分の位置を計算することで，多少の高速化を図った．さらに高速に H を作成するには，H が Pauli 行列と単位行列のテンソル積の和で表されることから，このクロネッカー積の非零成分を右側の積から順に追っていくことにより $\mathcal{O}(N2^N)$ で非零成分の位置を求めることができるようである（[1] 参照）．

3　結果及び考察

以下では，反強磁性的な相互作用を考え，式 (1) における J を 1 とした．

$N = 3$ の場合の対角化

ハミルトニアン行列は以下のようになった．

```
[[ 0.75  0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -0.25  0.5   0.    0.5   0.    0.    0.  ]
 [ 0.    0.5  -0.25  0.    0.5   0.    0.    0.  ]
 [ 0.    0.    0.   -0.25  0.    0.5   0.5   0.  ]
 [ 0.    0.5   0.5   0.   -0.25  0.    0.    0.  ]
 [ 0.    0.    0.    0.5   0.   -0.25  0.5   0.  ]
 [ 0.    0.    0.    0.5   0.    0.5  -0.25  0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.75]]
```

また，この固有値は，

```
[-0.75 -0.75 -0.75 -0.75  0.75  0.75  0.75  0.75]
```

固有ベクトルは，

```
[[ 0. 0.78446454 -0.19611614 0. -0.58834841 0. 0. 0.]
```

```
[ 0. 0. 0. -0.78446454 0. 0.19611614 0.58834841 0.]
[-0. 0.22645541 -0.79259392 0. 0.56613852 0. 0. 0.]
[ 0. 0. 0. -0.22645541 0. 0.79259392 -0.56613852 0.]
[-0. -0. -0. 0.57735027 0. 0.57735027 0.57735027 0.]
[ 0. 0.57735027 0.57735027 0. 0.57735027 0. 0. 0.]
[ 1. 0. 0. 0. 0. 0. 0. 0.]
[ 0. 0. 0. 0. 0. 0. 0. 1.]]
```

となり，4 重縮退が 2 つ現れることが分かる．$N = 2$ の場合は全スピン演算子を $\boldsymbol{S}_{\rm tot} = \boldsymbol{S}_x + \boldsymbol{S}_y + \boldsymbol{S}_z$ として $H = \frac{J}{2}(\boldsymbol{S}^2_{\rm tot} - \frac{9}{4})$ と表せ，励起状態は $S_{\rm tot} = \frac{3}{2}$ の状態が 4 重，基底状態は $S_{\rm tot} = \frac{1}{2}, \frac{1}{2}$ の 2 つの状態がそれぞれ 2 重に縮退していることが分かる．

疎行列を用いた場合との計算時間の比較

次に密行列を用いた場合と，疎行列を用いた場合（コードをレポート末尾に示した）での基底状態のエネルギーの計算時間の比較を行った．また，この系では $S^z_{\rm tot}$ が保存することから，$S^z_{\rm tot}$ 毎のブロック行列で対角化を行うことで更なる効率化を図った（これも末尾にコードを示した）．以上の 3 つのやり方で，$N = 10$ の場合と $N = 13$ の場合の実行時間を比較したところ，以下のようになった（それぞれ 10 回ずつ計算して平均をとった）．

表 1　計算時間の比較

	密行列	疎行列	保存量
計算時間 [sec]（$N = 10$）	0.036 ± 0.002	0.019 ± 0.001	0.066 ± 0.006
計算時間 [sec]（$N = 13$)	2.15 ± 0.29	0.206 ± 0.049	0.487 ± 0.110

表 1 より，N が大きい場合は疎行列で扱う方が計算時間が短いことが分かる．実際，$N \geq 15$ とすると，密行列で扱うと時間が膨大にかかった．また表 1 を見ると，$S^z_{\rm tot}$ 毎のブロック行列で対角化を行った場合の計算時間が，単に疎行列で計算した場合より長くなっていることが分かる．これは，作成したコードではハミルトニアンを $S^z_{\rm tot}$ 毎の $2^N \times 2^N$ 行列の和にしそれぞれの対角化を行っただけで，小さい行列の対角化にしていないためであり，各ブロック行列を適切に求めることができればより高速になると考えられる．

N が大きいときの基底状態のエネルギー

1 次元 N サイトハイゼンベルグ模型は，ベーテ仮説による厳密解が知られており，$N \to \infty$ のもと $E/NJ = -\log 2 + 1/4$ （テキスト例題 13 より）である．まず，いくつかの N における基底状態のエネルギー（を N で割ったもの）を求めると，以下の表 2 のようになった．

表 2　基底状態のエネルギー E/N の値

N	2	5	10	12	15	20
E/N	-0.625	-0.374	-0.452	-0.449	-0.436	-0.445

続いて $N \to \infty$ での厳密解と数値解との比較を行うと以下の図 1 のようになった．表 2，図 1 より，N が大きくなるにつれ E/N は厳密解に収束することが確認できた．

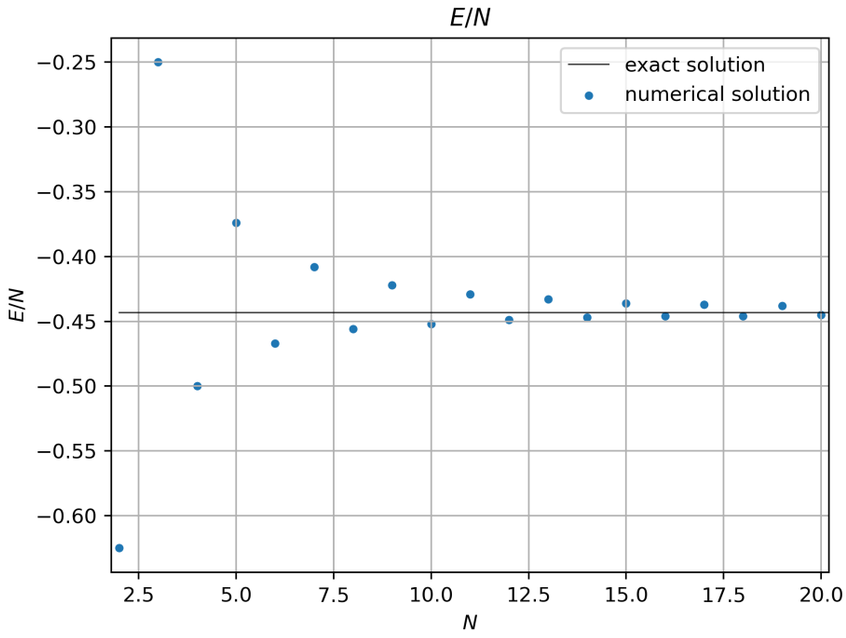

図 1　E/N の $N \to \infty$ での厳密解と数値解との比較

参考文献

[1] Qiita. ”量子スピン模型のハミルトニアンを数値対角化しよう”.
`https://qiita.com/fockl/items/52593901b9a5f78cda92`, (参照 2022-1-19).

付録

疎行列を用いた場合のコード

```
import numpy as np
from scipy.sparse import linalg,coo_matrix,csr_matrix

J = 1

def diag(i,n): #diagonal element
    a = "{:0>{}b}".format(i, n) #make binary (length n)
    count = int(a[0])^int(a[-1]) #count 10 or 01
    if n == 2:
        return (count-1/2)*(-1/2)
    else:
        for j in range(n-1):
            count += int(a[j])^int(a[j+1])
        return (n-2*count)*J/4

def offdiag_arg(i,n): #offdiagonal element (upper triangular)
    row = []
    col = []
    a = format(i,'b')
    l = len(a)
    for j in range(l-1): #swap 10 for 01
        if a[j] == "1" and a[j+1] == "0":
            row += [i-2**(l-j-1)+2**(l-(j+1)-1),i]
            col += [i,i-2**(l-j-1)+2**(l-(j+1)-1)]
    if l == n and a[-1] == "0": #swap first 1 and last 0
        row += [i-2**(l-1)+1,i]
```

```
            col += [i,i-2**(l-1)+1]
        return row, col

    def hamiltonian(n): #make hamiltonian matrix
        data = []
        row = []
        col = []
        for i in range(2**n):
            data.append(diag(i,n))
            row.append(i)
            col.append(i)
            r,c = offdiag_arg(i,n)
            l = len(r)
            data += [J/2 for _ in range(l)]
            row += r
            col += c
        H_coo = coo_matrix((data,(row,col)),shape=(2**n,2**n))
        H_csr = csr_matrix(H_coo)
        return H_csr
```

S^z_{tot} ごとの空間で対角化を行う場合のコード

```
    def hamiltonian(n):
    H = [[],[],[]] for i in range(n+1)] #lists for each num of |-> spins
    for i in range(2**n):
        a = "{:0>{}b}".format(i, n) #make binary (length n)
        l = len(a)
        a_array = np.array([int(a[i]) for i in range(l)])
        Sz = int(np.dot(a_array,np.ones(l))) # num of |-> spins
        data = [diag(i,n)] #diagonal element
        row = [i]
        col = [i]
        r,c = offdiag_arg(i,n) #offdiagonal element
        l = len(r)
        data += [J/2 for _ in range(l)]
        row += r
        col += c
        H[Sz][0] += data
        H[Sz][1] += row
        H[Sz][2] += col
    return H

    def ground(H,n): #energy of ground state
    eig_gr = H[0][0]
    for i in range(1,n+1):
        data,row,col = H[i][0],H[i][1],H[i][2]
        H_coo = coo_matrix((data,(row,col)),shape=(2**n,2**n))
        eig,vec = linalg.eigs(H_coo, k=1)
        if eig < eig_gr:
            eig_gr = eig
    return eig_gr
```