Titpack—Numerical diagonalization routines and quantum spin Hamiltonians

Hidetoshi Nishimori

Citation: AIP Conference Proceedings **248**, 269 (1992); doi: 10.1063/1.41947
View online: http://dx.doi.org/10.1063/1.41947
View Table of Contents:
http://scitation.aip.org/content/aip/proceeding/aipcp/248?ver=pdfcov
Published by the AIP Publishing

TITPACK
— NUMERICAL DIAGONALIZATION ROUTINES OF QUANTUM SPIN HAMILTONIANS —

Hidetoshi Nishimori
Department of Physics, Tokyo Institute of Technology, Oh-okayama, Meguro-ku, Tokyo 152, Japan

ABSTRACT

The computer program package TITPACK Ver.2 has been designed to facilitate numerical calculations of the low-lying energy levels of quantum spin systems. For spin-1/2 models, it diagonalizes the Hamiltonian numerically for the specified values of parameters (system size, anisotropy, lattice structure etc.) as long as the memory and CPU time requirements are satisfied on the user's computer. Several low-lying energy eigenvalues and eigenvectors are returned automatically by calling a few subroutines. The basic algorithm is the well-known Lanczos method.

INTRODUCTION

The ground states of quantum spin systems do not have trivial structures if the exchange interactions are antiferromagnetic. A simple example can be found in the two-spin system with spin size 1/2, in which the singlet ground state is quite different from the classical Neel state. The situation is much more complicated in thermodynamically large systems. Exact solutions are found for the one-dimensional case with uniform antiferromagnetic interactions. Most of the other systems escape rigorous analysis in spite of recent high interest in the two-dimensional Heisenberg antiferromagnet.

Numerical approaches often show their power when reliable results are not easily obtained by analytic techniques. One of the main numerical methods in investigating quantum spin systems is the quantum Monte Carlo simulations pioneered by the Suzuki group.[1] This method has an advantage that any system can be treated in principle and main memory requirement is not generally large. However, as is usual for any Monte Carlo simulations, the problem of statistical errors is sometimes serious, in particular when one treats frustrated systems. Very large, often inhibitingly large, CPU time is required to obtain significant data for frustrated cases such as the triangular lattice model and the antiferromagnet with next nearest neighbor interactions.

Another technique is the direct numerical diagonalization of spin Hamiltonian matrices, which became rather popular recently partly because of the fast development of hardware technology of computers. The leading advantage of the diagonalization method is the absence of statistical errors. The method is useful also because the necessary CPU time is not large, often negligible, as compared with Monte Carlo simulations. One does not have to apply for a large grant to carry out a research program using numerical diagonalization. However, the memory requirement is not small, which sets a strong limit on the size of systems to be treated by numerical diagonalization. Therefore one has to determine carefully what method to use in investigating quantum spin systems by numerical calculations. TITPACK is a useful tool for those who wish to check their physical intuition by numerical diagonalization of finite-size systems.

HISTORICAL BACKGROUND

The system to be treated by the package TITPACK is specified by the Hamiltonian

$$H = -2 \sum_{i,j} J_{ij} (S_i^x S_j^x + S_i^y S_j^y + \Delta_{ij} S_i^z S_j^z) . \quad (1)$$

The spin size S is 1/2. The parameters J_{ij} and Δ_{ij} can be arbitrarily set by the user. In particular, the lattice structure is specified by appropriately setting the values of exchange interactions.

The first version of the package, written by Y. Taguchi and the present author, was released in 1986.[2] The program was intended to be used on a Hitachi's supercomputer HITAC S-810/20 at the Computer Center of the University of Tokyo. This machine had the theoretical peek speed of 800 MFLOPS and the maximum main memory space accessible for a general user was 32 MBytes at that time. This specification enabled us to diagonalize a 21-spin system without using translational symmetries; we were investigating the problem of triangular lattice antiferromagnet by the railroad trestle extrapolation method,[3] which did not allow lattice symmetry other than the trivial conservation of total S_z.

This first version has since been used by many researchers in the field of quantum spin systems in Japan. The main reasons for not having been used abroad were that the program used specific library calls of HITAC and that the specifications for automatic vectorization of FORTRAN codes were different on Japanese supercomputers from those of foreign machines. We therefore did not write a manual in English.

Hardware of computers has advanced quite significantly in these several years. In 1988, a new supercomputer HITAC S-820/80 was introduced to the University of Tokyo Computer Center. This machine has the peek speed of 3GFLOPS and the current maximum memory per user is 480 MBytes. A new supercomputer, Fujitsu's FACOM VP-2600, was installed in 1991 at the Data Processing Center of Kyoto University. Its speed is 5 GFLOPS at the theoretical peek performance, and the main memory for a general user is available up to 200 MBytes. Situations in smaller machines have also changed, and it is now rather easy to install a workstation in one's own office.

The new version of TITPACK has increased portability to catch up with these changes of situations. It runs without library calls of a specific machine since it is written using only FORTRAN 77 statements with the single exception of bit-wise logical operations of two integers, IAND (logical And of corresponding bits) and IEOR (logical Exclusive Or of bits). Since these two functions are now supported by most FORTRAN compilers, in this form or another, this would not set a serious difficulty in using TITPACK on various machines.

USAGE

To use TITPACK Ver. 2, the user has first to specify various parameters in the Hamiltonian (1). With these parameters given in data or other statements, he/she calls several routines successively to get eigenvalues, eigenvectors and correlation functions. An example is given below in which one calculates lowest eigenvalues, an eigenvector and nearest neighbor correlation functions of the one-dimensional Heisenberg antiferromagnet with n(the number of spins)=16. The reader is referred to the manual, which is available from the present author upon request, for detailed explanation of routines; I only give a short account here. The routine `sz` generates spin configurations in the space of given `n` and total S_z. The four lowest eigenvalues are calculated in `lnc1`, and the eigenvector in `lncv1`. The precision of the eigenvalues and eigenvector is checked in `check1`, and the correlation functions are evaluated in `xcorr` and `zcorr`.

```
c************* Sample program #1 ***************
c   Eigenvalues and an eigenvector / lnc1, lncv1
c    Precision check and correlation functions
c***********************************************
      parameter (n=16,idim=12870,ibond=n)
      implicit real*8 (a-h,o-z)
      dimension E(4)
      dimension list1(idim),list2(2,0:2**15)
      dimension bondwt(ibond),ipair(2*ibond),zrtio(ibond)
      dimension npair(2)
      dimension wk(idim,2)
      dimension x(idim)
c
      data bondwt/ibond*-1.0d0/
      data zrtio/ibond*1.0d0/
      data ipair/1,2, 2,3,  3,4,  4,5,  5,6,  6,7,  7,8, 8,9,
     &     9,10, 10,11, 11,12, 12,13, 13,14, 14,15, 15,16, 16,1/
      nvec=1
      iv=idim/3
      call sz(n,idim,0.0d0,list1,list2)
c
c*** Eigenvalues
      call lnc1(n,idim,ipair,bondwt,zrtio,ibond,
     &                 nvec,iv,E,itr,wk,idim,list1,list2)
      print 100,e,itr
 100  format(/' [Eigenvalues]  '/2x,4f14.8
     &       /' [Iteration number]'/i8)
c
c*** Ground-state eigenvector
      call lncv1(n,idim,ipair,bondwt,zrtio,ibond,
     &             nvec,iv,x,itr,wk,idim,list1,list2)
```

```
      print *,'[Eigenvector components (selected)]'
      print 120,(x(j),j=13,idim,idim/20)
 120  format(4d18.9)
c
c*** Precision check and correlation functions
      call check1(n,idim,ipair,bondwt,zrtio,ibond,
     &            x,wk,Hexpec,list1,list2)
      npair(1)=1
      npair(2)=2
      call xcorr(n,idim,npair,1,x,sxx,list1,list2)
      call zcorr(n,idim,npair,1,x,szz,list1)
      print 130,sxx,szz
 130  format(/' [Nearest neighbor correlation functions]'/
     &       '    sxx :',d18.10,',    szz :',d18.10)
      end
```

```
==================== RESULT 1 =====================

[Eigenvalues]
   -14.28459272  -13.74421336  -13.39309484  -13.04681410
[Iteration number]
     50
[Eigenvector components (selected)]
   0.343792928D-05   0.125778312D-01   0.892952949D-04  -0.216465016D-02
  -0.341393531D-02   0.717748830D-04  -0.346965348D-02   0.124963114D-03
  -0.121707616D-02  -0.341393530D-02   0.338263590D-07   0.391932543D-03
   0.266317018D-02  -0.431510412D-03   0.392280336D-02  -0.113573424D-03
   0.578613958D-02   0.448821353D-02   0.316356482D-06   0.169426150D-03

---------------------------- Information from check1
<x*H*x> = -1.42845927D+01
H*x(j)/x(j) (j=min(idim/3,13),idim,max(1,idim/20))
  -0.142845090D+02  -0.142845927D+02  -0.142845925D+02  -0.142845927D+02
  -0.142845927D+02  -0.142845924D+02  -0.142845927D+02  -0.142845936D+02
  -0.142845927D+02  -0.142845927D+02  -0.142825155D+02  -0.142845931D+02
  -0.142845927D+02  -0.142845932D+02  -0.142845927D+02  -0.142845920D+02
  -0.142845927D+02  -0.142845927D+02  -0.142844308D+02  -0.142845933D+02
---------------------------------------------------

[Nearest neighbor correlation functions]
   sxx : -0.1487978408D+00,    szz : -0.1487978407D+00
```

Three routine groups are included in TITPACK Ver. 2, L, M and S. The user chooses an appropriate group for his/her purposes. A rough criterion is the size of the system to be diagonalized. The group L is for large systems, M for medium and S for small. Optimal performance is expected by using the best group for the size. The above sample program uses group L although group M is usually appropriate for this size on most computers. The reader is referred to the manual for details. Examples of CPU time and memory requirements are presented in the following section for each of these groups.

I have to add here that no reduction of matrix size by lattice symmetry is implemented, except for the trivial classification by total S_z, because this package is intended for general use including random system. One therefore has to rewrite relevant parts of the source code if he/she wishes to take full advantage of symmetries of a specific lattice. I think that this simple program without elaboration on symmetry reduction is usually sufficient to get a first-hand test of one's physical intuition, which is actually how the first version has been used.

BENCHMARK TEST

Tables I to III list the necessary main memory space and CPU time on HITAC S-820/80 and NEC EWS-4800/220 (a work station). The Hamiltonian is the one-dimensional Heisenberg antiferromagnet with nearest neighbor interactions. The four lowest eigenvalues in the space $S_z^{\text{total}} = 0$ and the ground-state eigenvector have been calculated. All inner do loops have been automatically vectorized by the compiler on HITAC S-820/80.

Table I Main memory requirement
(in MB; – : less than 1MB; * : exceeding 1GB)

n	matrix size	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
12	924	–	–	6.6	–	–	–	–	6.6
16	12,870	–	2.7	*	–	–	2.8	3.0	*
20	184,756	3.5	48	*	4.9	7.8	49	51	*
24	2,704,156	52	701	*	72	113	836	877	*
26	10,400,600	198	*	*	277	436	*	*	*

Table II CPU time on S-820/80 (sec)
(– : under 1 sec; * : unexecutable for insufficient memory)

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
12	–	–	–	2	–	–	–	–	2
16	–	–	–	*	–	1	–	–	*
20	5	10	1	*	20	38	2	4	*
24	80	225	*	*	448	839	*	*	*
26	332	883	*	*	1758	2780	*	*	*

Table III CPU time on EWS-4800/220 (sec)
(– : under 1 sec; * : unexecutable for insufficient memory)

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
12	–	1	1	816	2	4	2	3	816
16	5	34	21	*	68	128	41	95	*
20	104	852	*	*	1710	2988	*	*	*

I	: Eigenvalues by Group L
II	: Eigenvalues by Group M
III	: Eigenvalues by Group S
Iv	: Eigenvalues and an eigenvector by Group L
Iv′	: Eigenvalues and an eigenvector by Group L
IIv	: Eigenvalues and an eigenvector by Group M
IIv′	: Eigenvalues and an eigenvector by Group M
IIIv	: Eigenvalues and an eigenvector by Group S

BASIC ALGORITHM

Routines in groups L and M use the Lanczos method to tridiagonalize a matrix and the inverse iteration method combined with the Conjugate Gradient method to calculate eigenvectors. I give short accounts of these methods. The reader is referred to Ref. 4 and other textbooks of numerical techniques for details.

It is instructive first to explain the Lanczos method as an improvement of the power method. The power method is a simple way to calculate the eigenvalue of a matrix with the largest absolute value; one starts from an arbitrary initial vector $\mathbf{v}_0$ and multiplies the matrix H repeatedly until the resulting vector converges to the desired eigenvector. More precisely, when the eigenvalues and eigenvectors of an m-dimensional matrix H are $E_j, \psi_j (j = 1, \cdots, m)$, the expansion of the initial vector reads

$$\mathbf{v}_0 = \sum_{j=1}^{m} a_j \psi_j$$

Multiplying this initial vector k times by H, one obtains

$$\mathbf{v}_k \equiv H^k \mathbf{v}_0 = \sum_{j=1}^{m} a_j E_j^k \psi_j$$

It is apparent that the relative weight of the eigenvector corresponding to the eigenvalue with the largest absolute value increases exponentially with k among terms appearing in the above sum.

Acceleration of convergence over the simple power method is achieved by subtracting components of previous vectors $(\mathbf{v}_{k-1}, \mathbf{v}_{k-2}, \cdots)$ from $\mathbf{v}_k$ so that one can eliminate the effects of the arbitrarily chosen initial vector as rapidly as possible. This subtraction of components of previous vectors is incidentally equivalent to tridiagonalization. Let us explain this point.

If the tridiagonal matrix T is obtained from the original matrix H by a transformation matrix V, one has the relation $T = V^{-1}HV$, or $VT = HV$. Let the column vectors of V be $\mathbf{v}_1, \mathbf{v}_2, \cdots$, and the diagonal elements of T be $\alpha_1, \alpha_2, \cdots$, and the subdiagonal elements $\beta_1, \beta_2, \cdots$. Then the relation $VT = HV$ is written as

$$\begin{aligned} H\mathbf{v}_1 &= \alpha_1 \mathbf{v}_1 + \beta_1 \mathbf{v}_2 \\ H\mathbf{v}_2 &= \beta_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \beta_2 \mathbf{v}_3 \\ H\mathbf{v}_3 &= \beta_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \beta_3 \mathbf{v}_4 \\ &\cdots \\ H\mathbf{v}_{m-1} &= \beta_{m-2} \mathbf{v}_{m-2} + \alpha_{m-1} \mathbf{v}_{m-1} + \beta_{m-1} \mathbf{v}_m \\ H\mathbf{v}_m &= \beta_{m-1} \mathbf{v}_{m-1} + \alpha_m \mathbf{v}_m \end{aligned} \qquad (2)$$

If one rewrites (2) into a form to calculate $\mathbf{v}_k$ successively,

$$\begin{aligned}
\mathbf{v}_2 &= (H\mathbf{v}_1 - \alpha_1\mathbf{v}_1)/\beta_1 \\
\mathbf{v}_3 &= (H\mathbf{v}_2 - \beta_1\mathbf{v}_1 - \alpha_2\mathbf{v}_2)/\beta_2 \\
\mathbf{v}_4 &= (H\mathbf{v}_3 - \beta_2\mathbf{v}_2 - \alpha_3\mathbf{v}_3)/\beta_3 \\
&\cdots \\
\mathbf{v}_m &= (H\mathbf{v}_{m-1} - \beta_{m-2}\mathbf{v}_{m-2} - \alpha_{m-1}\mathbf{v}_{m-1})/\beta_{m-1}
\end{aligned} \tag{3}$$

The last equation in (3) corresponds the second last of (2). The requirement that the last relation of (2) is compatible with that of (3) leads to the vanishing value of $\mathbf{v}_{m+1}$ when (3) is formally extended to $m+1$. In order to have a vanishing $(m+1)$th vector in the series of $\mathbf{v}_k$, it is sufficient to choose $\mathbf{v}_k$ so that it is orthogonal to all previous vectors $\mathbf{v}_{k-1}, \mathbf{v}_{k-2}, \cdots$, since $(m+1)$ vectors cannot be orthogonal to each other in the m-dimensional space. It is useful here to regard (3) as an iterative orthogonalization process by subtracting components of previous vectors. Actually, it turns out that[4] by choosing

$$\begin{aligned}
\alpha_i &= \mathbf{v}_i^T H \mathbf{v}_i \\
\beta_i &= \| H\mathbf{v}_i - \beta_{i-1}\mathbf{v}_{i-1} - \alpha_i\mathbf{v}_i \|
\end{aligned} \tag{4}$$

all $\mathbf{v}_k$ are orthogonal to each other. Since this process (3) and (4) can be regarded as an improvement of the power method, it is not necessary to execute all of $(m-1)$ steps in (3) if one wishes to evaluate only several low-lying eigenvalues; one may calculate the eigenvalues of the intermediate tridiagonal matrix of dimension $l(< m)$ by the bisection method to see whether or not the low-lying eigenvalues have reached sufficiently converged values.

As for the eigenvectors, one first calculates the eigenvectors of the tridiagonal matrix when convergence of eigenvalues is confirmed. One then transforms the eigenvectors of the tridiagonal matrix into the original representation by use of the transformation matrix V. That is, the eigenvectors in the original representation are obtained by summing up the products of the components $c_1, c_2, \cdots$ of the eigenvector in the tridiagonal representation and the column vectors $\mathbf{v}_1, \mathbf{v}_2, \cdots$ of V: $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 \cdots$. However, it is difficult to store the sequence generated by (3) $\mathbf{v}_1, \mathbf{v}_2, \cdots$ until convergence since convergence usually results after tens of iterations (and tens of the vectors should be stored in memory). A simple way out is to store $c_1, c_2, \cdots$ (real numbers) in the first run, and repeat the Lanczos process to generate $\mathbf{v}_1, \mathbf{v}_2, \cdots$ and sum up the products of the vectors and $c_1, c_2, \cdots$.[5] This method is used in some routines.

Inverse iteration is a method to calculate the eigenvector corresponding to a given approximate eigenvalue E_a. One starts from an arbitrary initial vector $\mathbf{v}_0$ and repeatedly multiplies $(H - E_a)^{-1}$. Let the expansion of the initial vector by the eigenvectors of H be

$$\mathbf{v}_0 = \sum_{j=1}^{m} a_j \psi_j$$

Then, k multiplications of $(H - E_a)^{-1}$ lead to

$$\mathbf{v}_k \equiv (H - E_a)^{-k}\mathbf{v}_0 = \sum_{j=1}^{m} a_j (E_j - E_a)^{-k}\psi_j$$

This equation indicates that the contribution of the eigenvector corresponding to the eigenvalue closest to E_a becomes exponentially large as k increases. Convergence is faster for a better approximate eigenvalue; in TITPACK Ver. 2, the iteration usually converges after a few steps.

If one tries to directly execute multiplication of $(H - E_a)^{-1}$, one has to obtain the inverse matrix, which is in general a difficult task. However, the relation

$$\mathbf{v}_k = (H - E_a)^{-1}\mathbf{v}_{k-1}$$

is equivalent to

$$(H - E_a)\mathbf{v}_k = \mathbf{v}_{k-1}$$

which may be regarded as a system of linear equations to calculate $\mathbf{v}_k$, given $\mathbf{v}_{k-1}$. Thus standard techniques can be used.

The Conjugate Gradient method solves linear equations of large scale by regarding the equation $M\mathbf{x} - \mathbf{b} = 0$ as the extremization of a bilinear form $f(\mathbf{x}') \equiv (\mathbf{r}, M^{-1}\mathbf{r})$ of the residual $\mathbf{r} \equiv \mathbf{b} - M\mathbf{x}'$, where $\mathbf{x}'$ is an approximate solution. Here (,) denotes inner product. To extremize $f(\mathbf{x}')$, one iteratively improves the approximate solution $\mathbf{x}'$ by changing $\mathbf{x}'$ to the direction with the steepest gradient on the landscape of $f(\mathbf{x}')$. The point where $f(\mathbf{x}')$ is extremum along this direction may be chosen as the next improved approximation. This is the Steepest Decent method. The Conjugate Gradient method is its improvement in that the direction along which a new approximation is sought is chosen within a space orthogonal to all previous modification vectors.

CONCLUSION

TITPACK Ver. 2 is a package of computer programs to numerically diagonalize quantum spin systems. It enables a researcher to obtain low-lying energy levels of any quantum spin system of the form in Eq. (1), as long as the computer to be used accommodates the necessary memory. It is sufficient to simply call several subroutines. It is an automatic vendor of eigenstates (Fig. 1); it requires no knowledge of algorithm of numerical diagonalization of large matrices. The user just enters the Hamiltonian and pays for CPU time (some people may be able to skip this last part) to get results.

The source code is available from the author by e-mail. The address is A85085@JPNKUDPC.BITNET. The source may also be obtained by mail in an IBM-PC formatted 5-inch diskette. The manual will be mailed upon request. There is no restriction on personally copying and using TITPACK Ver. 2 for academic purposes. I only ask the user to acknowledge its use, as it is or

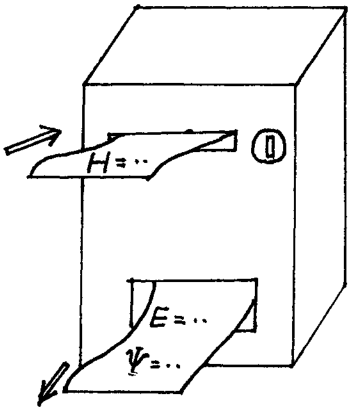

Fig. 1. TITPACK Ver. 2 as a vendor of eigenstates.

after modifications, with the present author's name explicitly indicated when publishing results.

ACKNOWLEDGMENT

I express my sincere thanks to Dr. Y. Taguchi, the co-author of Ver. 1. Useful suggestions on coding techniques by Dr. M. Ogata and Mr. M. Matsushita are gratefully acknowledged. This research was supported by the Grant-in-Aid for Priority Area by the Ministry of Education, Science and Culture.

REFERENCES

1. M. Suzuki, in *Quantum Monte Carlo Methods in Equilibrium and Nonequilibrium Systems* (Springer, Berlin, 1987).
2. Y. Taguchi and H. Nishimori, Bussei Kenkyu **45**, 299 (1986) (in Japanese). H. Nishimori and Y. Taguchi, Prog. Theor. Phys. Suppl. No. 87, 247 (1986).
3. T. Oguchi, H. Nishimori and Y. Taguchi, J. Phys. Soc. Jpn. **55**, 323 (1986).
4. H. Togawa, *Numerical Calculations Of Matrices* (in Japanese) (Ohm Sha, Tokyo, 1971).
5. E.R. Gagliano, E. Dagotto, A. Moreo and F.C. Alcaraz, Phys. Rev. B **34**, 1677 (1986). See also *Encyclopedia of Mathematical Sciences,* Ed. H. Hironaka, (Maruzen, Tokyo, 1991) (in Japanese), p. 909 for detailed analysis of the precision of eigenvectors.