Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 4512–4517
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Efficient implementation of the Lanczos method for magnetic systems

Jürgen Schnack [a,*], Peter Hage [b], Heinz-Jürgen Schmidt [b]

[a] Universität Bielefeld, Fakultät für Physik, Postfach 100131, D-33501 Bielefeld, Germany
[b] Universität Osnabrück, Fachbereich Physik, D-49069 Osnabrück, Germany

Received 22 June 2007; received in revised form 11 December 2007; accepted 10 January 2008
Available online 1 February 2008

Abstract

Numerically exact investigations of interacting spin systems provide a major tool for an understanding of their magnetic properties. For medium size systems the approximate Lanczos diagonalization is the most common method. In this article we suggest two improvements: efficient basis coding in subspaces and simple restructuring for openMP parallelization.

PACS: 75.10.Jm; 75.40.Mg

Keywords: Spin systems; Lanczos diagonalization; Basis coding; Parallelization

1. Introduction

Many magnetic materials can accurately be described by the Heisenberg or related effective spin models. Due to the vastly increasing size of the underlying Hilbert space, which grows as $(2s+1)^N$ for N spins of spin quantum number s, only small spin systems can be modeled exactly, i.e., their complete eigenspectrum can be determined. For larger systems approximate methods such as the Lanczos [1] or related methods like the Arnoldi, the projection, or the Density Matrix Renormalization Group (DMRG) method [2–4] are used. They usually aim at properties of ground states in orthogonal subspaces, which are provided by symmetry, see e.g. [5–7]. But also thermal properties can be addressed by means of a finite-temperature Lanczos method [8] as done for instance for the evaluation of certain Kondo lattice models in Ref. [9].

For all these methods it is of course advantageous to use the present symmetries in order to reduce the size of the Hamiltonian matrix as much as possible by decomposing the Hilbert space into mutually orthogonal subspaces. One obvious symmetry is the rotational invariance of many models with respect to rotations about the z-axis in spin space. This leads to a decomposition of the total Hilbert space $\mathcal{H}$ into orthogonal subspaces $\mathcal{H}(M)$ characterized by their total magnetic quantum number M. The related basis, which is a subset of the full

* Corresponding author. Tel.: +49 521 106 6193; fax: +49 521 106 6455.
E-mail address: jschnack@uni-bielefeld.de (J. Schnack).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.01.027

basis, should then efficiently be encoded. In many applications nowadays these basis states are either stored in tables and assessed via hash search methods, see e.g. [10], or encoded using the two-dimensional representation by Lin [11], which needs two vectors of size $\approx(2s+1)^{(N/2)}$ for encoding. In this article we will provide direct algorithms for encoding and decoding of basis states in subspaces $\mathcal{H}(M)$.

Thanks to available SMP (symmetric multiprocessing) computers with large shared memory Lanczos vectors of considerable size can be processed. An example is given in Ref. [7] where Lanczos vectors with about 10^9 entries were used. We show that by a simple reformulation of the typical implementation of the Lanczos algorithm a very sufficient parallelization with openMP can be achieved that avoids write conflicts.

The article is organized as follows. The next section shortly introduces the Heisenberg model as an archetypical example. In Section 3 we introduce the new basis encoding in subspaces $\mathcal{H}(M)$. The last Section 4 deals with parallelization issues.

2. Heisenberg Hamiltonian and basis encoding

Spin systems are very often modeled by effective spin Hamiltonian such as the isotropic Heisenberg Hamiltonian

$$\underset{\sim}{H} = -\sum_{u,v} J_{uv}\, \underset{\sim}{\vec{s}}(u) \cdot \underset{\sim}{\vec{s}}(v). \tag{1}$$

$\underset{\sim}{\vec{s}}(u)$ are the individual spin operators at sites u. J_{uv} are the matrix elements of the symmetric coupling matrix. In the following we will assume that all spin quantum numbers are equal, i.e., $s_1 = s_2 = \cdots = s_N = s$.

The starting point for any diagonalization is the product basis of the single-particle eigenstates of all $\underset{\sim}{s_z}(u)$

$$\underset{\sim}{s_z}(u)|m_1,\ldots,m_u,\ldots,m_N\rangle = m_u|m_1,\ldots,m_u,\ldots,m_N\rangle. \tag{2}$$

These states are sometimes called Ising states. They span the full Hilbert space and are used to construct symmetry-related basis states. For encoding purposes, and since m_u can be half-integer, they are usually rewritten in terms of quantum numbers $a_u = s - m_u$ instead of m_u, where $a_u = 0, 1, \ldots, 2s$. The number of basis states, i.e., the dimension of the full Hilbert space, is $\dim(\mathcal{H}) = (2s+1)^N$. The complete basis set $|a_1,\ldots,a_u,\ldots,a_N\rangle$ provides itself a natural encoding given by the number system with basis $(2s+1)$. To give an example, the basis of a system of 8 spins $s = 1$ can be completely and easily encoded using all 8-digit numbers where each digit can assume the values 0, 1, 2:

$$\begin{aligned}
&|0,0,0,0,0,0,0,0\rangle\\
&|1,0,0,0,0,0,0,0\rangle\\
&|2,0,0,0,0,0,0,0\rangle\\
&0,1,0,0,0,0,0,0\rangle\\
&\qquad\cdots\\
&|2,2,2,2,2,2,2,2\rangle.
\end{aligned} \tag{3}$$

3. Basis encoding in $\mathcal{H}(M)$

The basis in the subspace $\mathcal{H}(M)$ is given by all product states $|a_1,\ldots,a_N\rangle$ with $M = Ns - \sum_u a_u$. For usage in a computer program they need to be assigned to integer numbers $1,\ldots,\dim(\mathcal{H}(M))$. The reason is that one usually does not need the basis only once at initialization, but at every Lanczos iteration, since the sparse Hamiltonian matrix is not stored, but its non-zero matrix elements are evaluated whenever needed using

$$\langle i|\,\underset{\sim}{H}\,|j\rangle \equiv \langle a_1^i,\ldots,a_N^i|\,\underset{\sim}{H}\,|a_1^j,\ldots,a_N^j\rangle. \tag{4}$$

For a direct coding algorithm of basis states in subspaces $\mathcal{H}(M)$ it is advantageous that the sizes of the subspaces $\mathcal{H}(M)$ are known analytically [12]. Thus an array can be built at startup that contains for a fixed s the sizes of these subspaces $\mathcal{H}(M = Ns - A)$ for given N and A. We will call this array $D(N, A)$. It will be used to

determine the sequential number of a basis vector in $\mathcal{H}(M)$. The recursive buildup is performed using the following relation between the sizes of subspaces

$$D(N,A) = \sum_{k=0}^{2s} D(N-1, A-k), \tag{5}$$

with $D(N=1,\ A=0,1,\ldots,2s)=1$, $D(N,\ A=0)=1$, and $D(N,\ A=1)=N$. If $A \notin \{0,1,\ldots,2N\ s\}$ then $D(N,\ A)=0$.

3.1. $i \Rightarrow |a_1^i,\ldots,a_N^i\rangle$

One coding direction, $i \Rightarrow |a_1^i,\ldots,a_N^i\rangle$, which is the more trivial direction, can be realized in several ways. If the basis is not too big one simply generates all basis states of the subspace $\mathcal{H}(M)$ in lexicographical order, compare (3), and stores the quantum numbers a_k^i of the ith vector in an array. The generation can either be performed by running through all basis states (3) and sorting out those which comply with the condition $M = Ns - \sum_u a_u$ or by algorithms that generate only those basis states that obey the condition already.

A direct algorithm $i \Rightarrow |a_1^i,\ldots,a_N^i\rangle$ using the known dimensions of the subspaces $\mathcal{H}(M = Ns - A)$ could be realized as follows:[1]

```
      m=0
      Ak = A
      do k=N,2,-1
         do n=0,2*s
    if(i.le.(m+D(k-1,Ak-n+1))) then
       BasisVector(k) = n
       Ak = Ak - n
       goto 100
            else
       m = m + D(k-1,Ak-n+1)
            endif
         enddo
100      continue
      enddo
      BasisVector(1) = Ak
```

`BasisVector` contains the N entries a_k. This algorithm will be made clearer when we explain the inverse algorithm in Section 3.2.

Nevertheless, since a Lanczos routine would run through a state vector along the lexicographical order of basis states one would actually only need a function that generates for a given basis state the succeeding basis state. To understand how this works it is helpful to picture the basis states $|a_1^i,\ldots,a_N^i\rangle$ as distributions of exactly $A = \sum_u a_u$ balls in N boxes, where each box can contain at most $2s$ balls. Thus the lexicographically lowest state is given by the distribution where the boxes are filled sequentially starting with the leftmost box, i.e., entry number 1.

How does one advance from one basis state to the succeeding one?

(1) Find the leftmost position k for which the entry is nonzero and the next entry is less than $2s$. If such a position does not exist, then there is no succeeding basis state.

[1] The given code uses FORTRAN notation. Nevertheless, it can be easily transformed into C. One should only pay attention to the fact that field indices in FORTRAN start at 1 not at 0. Therefore, the definition of the second field index of D has been modified accordingly.

(2) Take one (ball) out of entry (box) k and add it to the next entry to the right, i.e., entry (box) with index $k+1$.
(3) Empty all entries (boxes) 1 to k and fill this content (these balls) into the entries (boxes) starting from the left in lexicographical order.

Take as an example for $N = 8$, $s = 3/2$, and $A = 6$ the state $|0,0,0,2,3,1,0,0\rangle$. Entry number $k = 5$ from the left is the first position to fulfill the first condition. One out of the 3 is put into $k = 6$ yielding 2 there. Then the content of entries $k = 1,\ldots,5$ is taken and filled into the entries starting from the left. This content is 4 in the present example. Three out of the four can be filled into entry number 1. The rest fits into entry number 2. Therefore, the resulting basis state is $|3,1,0,0,0,2,0,0\rangle$.

3.2. $|a_1^i,\ldots,a_N^i\rangle \Rightarrow i$

The inverse direction is actually the nontrivial one, since the basis vectors are only a subset of the full basis set (3). Therefore, for the latter coding direction search algorithms are employed, e.g. [10], or the two-dimensional representation of Lin [11] is used, which needs two vectors of size $\approx(2s+1)^{(N/2)}$ to encode all basis states.

The position of a basis vector $|a_1,\ldots,a_N\rangle$ in the lexicographically ordered list of vectors will be determined by evaluating how many vectors lay before this vector. For this purpose the known dimensions of the subspaces $\mathcal{H}(M = Ns - A)$ are used again. We explain this procedure with an instructive example. Assume we investigate a spin system with $N = 4$ and $s = 3/2$ in a subspace of $A = 6$, i.e., $M = 0$. Our example basis vector is $|1,0,2,3\rangle$. In the list of basis vectors all vectors fulfilling one of the following criteria are listed before the example vector, the respective dimensions will be added:

- Vectors with 0, 1, or 2 instead of 3 as the first (rightmost) figure: Their dimensions are $D(3,6)$, $D(3,5)$, and $D(3,4)$, respectively, since the condition that $\sum_i a_i = A$ must be fulfilled in total.
- Out of all vectors where the first figure is 3, those where the second figure is 0 or 1 are listed before, thus their respective dimensions of $D(2,3)$ and $D(2,2)$ must be added.
- This procedure continues until the last figure. In the present example this yields 0 for the third figure and simply 0 for the last figure.
- Thus the number of the present vector is given by the sum of the mentioned dimensions plus one.

In a computer program one can evaluate the position i of $|a_1,\ldots,a_N\rangle$ in the list of basis vectors according to

```
Ak = A
i = 1
do k=N,2,-1
   do n=0,BasisVector(k)-1
      i = i + D(k-1,Ak-n+1)
   enddo
   Ak = Ak - BasisVector(k)
enddo
```

`BasisVector` contains the N entries a_k. If the array of dimension $D(N,A)$ is properly initialized, i.e., the field value is zero for non-valid combinations of N and A, then the sum can be performed in a computer program without paying attention to the restrictions for the indices.

4. Parallel Lanczos implementation on SMP machines

Parallelization of the Lanczos or similar methods aims at a parallelization of the basic matrix-vector operations. This has been reported as being extremely difficult due to prohibitive communication costs [13,14]. In this section we show that parallelization is in principle possible if (1) the sparse matrix is not stored but matrix

elements are evaluated whenever needed and (2) the loops for matrix-vector multiplication are rearranged. The efficiency of the method will be discussed.

The basic step of a Lanczos or a similar method consists in the (repeated) application of the sparse matrix, i.e., the Hamiltonian, onto an initial trial vector

$$\langle i|\psi_2\rangle = \sum_{j=1}^{\dim(\mathcal{H}(M))} \langle i|\,\underset{\sim}{H}\,|j\rangle\langle j|\psi_1\rangle. \tag{6}$$

Here $\langle j|\psi_1\rangle$ are the entries of the initial column vector $|\psi_1\rangle$; the resulting vector is $|\psi_2\rangle$.

(1) The Hamiltonian matrix $\langle i|\,\underset{\sim}{H}\,|j\rangle$ is sparse, both on the global level as well as in subspaces $\mathcal{H}(M)$ since for many applications the interactions given by J_{uv} in Eq. (1) are restricted to nearest neighbors. In such a subspace it typically contains an order of $N^{1\ldots2} \times \dim(\mathcal{H}(M))$ non-zero entries, for instance for Heisenberg systems. For very large dimensions, e.g. of order 10^9, this would easily amount to several dozens of Gigabytes. Therefore, it would be better not to store the matrix, but to evaluate the matrix elements whenever needed. In addition one saves communication time since the required matrix elements do not need to be delivered to the respective cores.

(2) A typical implementation would have the loop about j as the outermost loop. An entry $\langle j|\psi_1\rangle$ of the initial vector would be read, then the non-zero matrix elements $\langle i|\,\underset{\sim}{H}\,|j\rangle$ would be determined, and the resulting products would be written into the respective entries $\langle i|\psi_2\rangle$. When parallelizing the loop about j this leads to write conflicts since different initial entries may result in the same final one.

It turns out that both problems can be solved together in cases where the application of the Hamiltonian onto each basis state is known analytically. In these cases only the non-vanishing matrix elements will be generated by applying the Hamiltonian, e.g. (1), onto the *final* basis state $|i\rangle$. This yields for a given final index i a set of initial indices $\{j(i)\}$ where only these indices contribute in the sum in Eq. (6).

$$\langle i|\psi_2\rangle = \sum_{\{j(i)\}} \langle i|\,\underset{\sim}{H}\,|j\rangle\langle j|\psi_1\rangle. \tag{7}$$

Therefore, one would rewrite Eq. (6) as Eq. (7) and in a parallel computer program let i be the outer loop. Then one determines for every final entry $\langle i|\psi_2\rangle$ those initial entries $\langle j|\psi_1\rangle$ that contribute with non-zero $\langle i|\,\underset{\sim}{H}\,|j\rangle$ in the sum (7). It may happen that at runtime different threads read the same entry of the initial vector, but this is harmless.

Fig. 1 shows as an example the scaling of CPU time for 200 Lanczos iterations with a vector of length 484,500. The program and all subroutines are written in Fortran and compiled with the INTEL Fortran compiler using openMP directives. The linear scaling is almost perfect. Slight deviations are due the non-parallel parts of the program, especially the initialization.

Although this result is encouraging there are potential problems one still has to be aware of. In the tested SMP machine all eight cores are sitting on one and the same board together with the shared memory. How

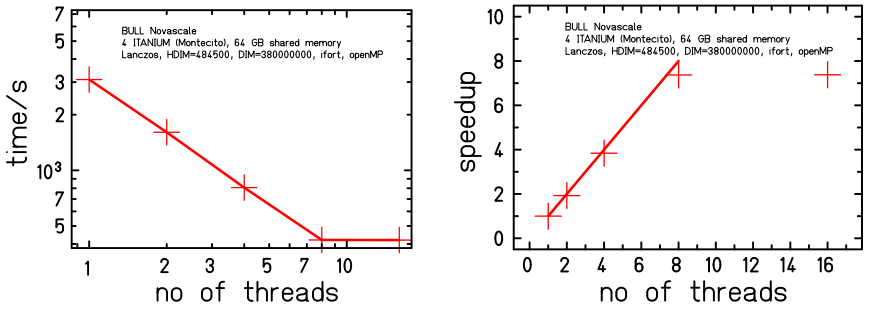

Fig. 1. Scaling of CPU time for 200 Lanczos iterations with number of allowed threads. The machine has eight cores.

good the algorithm scales, if one would use an architecture where cores are distributed over several boards with a necessarily increased communication between the boards, has to be tested. That communication will indeed be needed has to do with the fact that a Hamiltonian of type (1) connects very different parts of the Lanczos vector, in some cases it can even be proven that it connects every part with every other after a certain number of Lanczos steps. Nevertheless, in some cases it is possible to regroup basis states in order to minimize communication costs.

Summarizing, in this article we provide a coding algorithm for spin basis states in subspaces $\mathcal{H}(M)$ and demonstrate that a rearrangement of loops allows an efficient parallelization of the Lanczos algorithm. The proposed improvements can easily be ported to similar methods such as Arnoldi or projection method.

Acknowledgement

We thank J. Schulenburg and B. Schmidt for discussing their Lanczos implementations with us. We also thank J. Richter for drawing our attention to the encoding of H.Q. Lin, J.S. thanks M. Brüger and R. Schnalle for discussing encoding problems with him on a train ride.

References

[1] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Stand. 45 (1950) 255–282.
[2] S.R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B 48 (1993) 10345.
[3] S.R. White, D. Huse, Numerical renormalization-group study of low-lying eigenstates of the antiferromagnetic $s = 1$ heisenberg chain, Phys. Rev. B 48 (1993) 3844.
[4] U. Schollwöck, The density-matrix renormalization group, Rev. Mod. Phys. 77 (2005) 259–315.
[5] J. Schulenburg, A. Honecker, J. Schnack, J. Richter, H.-J. Schmidt, Macroscopic magnetization jumps due to independent magnons in frustrated quantum spin lattices, Phys. Rev. Lett. 88 (2002) 167207.
[6] J. Schnack, H. Nojiri, P. Kögerler, G.J.T. Cooper, L. Cronin, Magnetic characterization of the frustrated three-leg ladder compound $[(cucl_2tachh)_3cl]cl_2$, Phys. Rev. B 70 (2004) 174420.
[7] C. Schröder, H.-J. Schmidt, J. Schnack, M. Luban, Metamagnetic phase transition of the antiferromagnetic heisenberg icosahedron, Phys. Rev. Lett. 94 (2005) 207203.
[8] J. Jaklič, P. Prelovšek, Finite-temperature properties of doped antiferromagnets, Adv. Phys. 49 (2000) 1–92.
[9] I. Zerec, B. Schmidt, P. Thalmeier, Kondo lattice model studied with the finite temperature lanczos method, Phys. Rev. B 73 (2006) 245108.
[10] E.R. Gagliano, E. Dagotto, A. Moreo, F.C. Alcaraz, Correlation functions of the antiferromagnetic heisenberg model using a modified lanczos method, Phys. Rev. B 34 (3) (1986) 1677–1682.
[11] H.Q. Lin, Exact diagonalization of quantum-spin models, Phys. Rev. B 42 (1990) 6561–6567.
[12] K. Bärwinkel, H.-J. Schmidt, J. Schnack, Structure and relevant dimension of the Heisenberg model and applications to spin rings, J. Magn. Magn. Mater. 212 (2000) 240.
[13] R. Geus, S. Rollin, Towards a fast parallel sparse symmetric matrix-vector multiplication, Parallel Comput. 27 (2001) 883–896.
[14] W.W. Chen, B. Poirier, Parallel implementation of efficient preconditioned linear solver for grid-based applications in chemical physics. ii: QMR linear solver, J. Comput. Phys. 219 (2006) 198–209.