TITPACK Ver. 2

操 作 説 明 書

東京工業大学　大学院理工学研究科　物性物理学専攻

西 森 秀 稔

まえがき

このマニュアルは S = 1=2 の量子スピン系のハミルトニアンを数値的に対角化するためのプログラムパッケージ TITPACK の Ver. 2 の解説である．1985 年に発表した Ver. 1 は多くの人によって量子スピン系の研究に使われてきたが，その後のハードウェアの進歩や新しいアルゴリズムの導入をふまえて，このたび大幅な書換えを行い機能の向上を実現した．Ver. 2 の開発の指針としては，単により速く，より少ない記憶容量で走るようにというだけでなく，ワークステーション等による分散処理の進展にも対応するべく，異機種間の移植性を高めることにも留意した．そのために，プログラム内で大型計算機センターのライブラリー・プログラムを呼び出すことをやめ，必要な処理はすべて本パッケージ内で行なうようにした．TITPACK Ver. 2 の活用によって量子スピン系の性質の解明が一層進むことを願っている．

Ver. 2 の開発に当たっては，Ver. 1 の発表以来たくさんの人々から寄せられた有益な助言が役に立った．特に，小形正男氏 (東大物性研)，松下操氏 (千葉大理，現 CSK 総合研究所)，それに Ver. 1 の共同開発者の田口善弘氏 (東工大理) に感謝する．また，参考文献 2 のプログラムを部分的に使用することを許して下さった森正武氏 (東大工) と岩波書店にもお礼を申し上げる．

1991 年 3 月

西森秀稔

nishi@stat.phys.titech.ac.jp

目 次

§1. 概説

TITPACK Ver.2 は，次のハミルトニアンで表される $S = 1/2$ の系の基底状態およびその上の 3 つの励起状態を，数値的対角化の方法で有限の大きさの系について求めるプログラムである．

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

縮退がある場合にはそれも含めて数えて，低い順に 4 つの固有値と固有ベクトルが求められる．格子の形や相互作用のレンジ等は，J_{ij} を適当に与えることによって任意に設定できる．系の並進対称性や点群の対称性などによる行列の縮約は行っていない．これは，ランダム系を含むあらゆる問題に対応できるだけの汎用性を持たせるためである．したがって，個々のユーザが抱えている問題の特性を利用して現在の計算機の能力をぎりぎりまで引き出した計算をしたいと望むならば，自らプログラムを組む必要がある．その場合でも本プログラム・パッケージに盛り込まれたさまざまな工夫は役に立つことがあるかと思う．

各サブルーチンは $S_z^{\rm total}$ が正または 0 の一定の値をとる部分空間で (1) 式のハミルトニアンを対角化し，固有値，固有ベクトルを求める．

TITPACK Ver. 2 は FORTRAN77 の標準仕様に従って書かれてある．ただし，2 つの整数 i, j (いずれも 4 バイト) のビットごとの論理積と排他的論理和を取った結果を返す関数 IAND(i, j), IEOR(i, j) を使用しているので，これらがサポートされてない処理系では走らない．また，これらが別の表現 (例えば XOR など) を取る場合にはその部分を書換えないといけない．

1.1 TITPACK Ver. 2 に含まれるサブルーチンとそれぞれの機能

大規模行列用 (行列要素を求めながら対角化する)

サブルーチン名	およその機能
lnc1	Lanczos 法による固有値の計算
lncv1	Lanczos 法による固有ベクトルの計算
inv1	逆反復法による固有ベクトルの計算
check1	固有値，固有ベクトルの精度チェック

中規模行列用 (0 でない行列要素をあらかじめ求めてから対角化する)

サブルーチン名	およその機能
elm2	行列要素の計算
lnc2	Lanczos 法による固有値の計算
lncv2	Lanczos 法による固有ベクトルの計算
inv2	逆反復法による固有ベクトルの計算
check2	固有値，固有ベクトルの精度チェック

小規模行列用 (すべての行列要素をあらかじめ求めてから対角化する)

サブルーチン名	およその機能
elm3	行列要素の計算
diag	Householder 法と逆反復法による固有値，固有ベクトルの計算
check3	固有値，固有ベクトルの精度チェック

以上 3 つに共通して使われるもの

サブルーチン名	およその機能
sz	スピン配列の生成
xcorr	S_x, S_y の 2 点相関関数の計算
zcorr	S_z の 2 点相関関数の計算
orthg	複数ベクトルの直交化による縮退度の計算

以上の各ルーチン内で呼び出されている下位ルーチン (ユーザは通常意識しなくてよい)

サブルーチン名	およその機能
lnc1z	Lanczos 法による対角化の実行 (固有値)
lncv1z	Lanczos 法による対角化の実行 (固有ベクトル)
mltply	行列，ベクトルの積和
inv1z	逆反復法の実行
cg1	CG 法による連立方程式の解
lnc2z	Lanczos 法による対角化の実行 (固有値)
lncv2z	Lanczos 法による対角化の実行 (固有ベクトル)
inv2z	逆反復法の実行
cg2	CG 法による連立方程式の解
hshldr	Householder 法による 3 重対角化
vec3	逆反復法と LU 分解による固有ベクトルの計算
datack	サイトデータのエラーチェック
bisec	バイセクション法による 3 重対角行列の固有値
vec12	逆反復法と LU 分解による 3 重対角行列の固有ベクトルの計算

1.3 節以降で，これらのルーチン群が互いにどのように関連しあって全体の処理が行なわれるかを図示する．

1.2 ルーチン群の選択

大規模用，中規模用，小規模用のルーチン群を問題に応じてどう使い分けたら良いかの基準を示す．一般的に言うと，行列次元がおよそ 100 次元以下では小規模用にする．それ以上では，主記憶に 0 でない行列要素の情報が蓄えられる限りは中規模用が計算速度の点で大規模用にまさる．中規模用のルーチンで必要な記憶容量は下図にあるように，およそ

$$(\mathrm{idim} * \mathrm{ibond} * 1.5 * 8)/1024^2 \quad (\mathrm{MB})$$

である．idim は行列次元で，次のようにして求まる．スピン数 n と S_z^{total} が与えられたとする．上向きスピンの数を iu，下向きを id とすると iu+id=n, (iu−id)/2=S_z^{total}，これより iu=n/2−S_z^{total} となり，行列次元は n から iu を選ぶ組み合わせの数 idim=${}_{\mathrm{n}}\mathrm{C}_{\mathrm{iu}}$ と計算される．ibond は 0 でないボンド (J_{ij}) の数を表わす．

1.4 中規模行列における処理の流れ

1.6 ベンチマークテスト

それぞれのルーチン群で必要な記憶容量と，HITAC S-820/80，FACOM VP-2600，HITAC M-880/310，および NEC EWS-4800/220 での CPU time の一覧表を掲げる．調べた系は，1 次元反強磁性ハイゼンベルク模型で，最近接格子間にのみ相互作用があるものである．$S_z^{\rm total}=0$ の空間で基底固有値を含む 4 つの固有値と基底固有ベクトルを，初期条件 iv を idim/3 に取って求めた (idim は行列の次元，また iv の詳しい意味については 2.1 節の lnc1 の項を参照)．いずれもコンパイラによる最適化は最大限行っている．S-820/80 と M-880/310 の比較からわかるように，中規模用ルーチンは大規模用ルーチンに比べてベクトルプロセッサをより効率的に利用している．

他の系 (2 次元系など) の計算で必要な記憶容量は，大ざっぱに言って大規模用ルーチンでは idim のみに比例するが，中規模用ルーチンでは idim とボンド数 ibond の積に比例し，小規模用ルーチンでは idim の 2 乗に比例する．CPU time については，大規模用，中規模用ルーチンでは idim と ibond の積に比例し，小規模用ルーチンでは idim の 3 乗に比例する．

凡例

I　: 大規模行列用ルーチン lnc1 を使って固有値のみを求める場合
II　: 中規模行列用ルーチン lnc2　〃
III : 小規模行列用ルーチン diag　〃
Iv　: 大規模行列用ルーチン lnc1 と lncv1 使って固有値・固有ベクトルを求める場合
Iv′ : 大規模行列用ルーチン lnc1 と inv1　〃
IIv : 中規模行列用ルーチン lnc2 と lncv2　〃
IIv′ : 中規模行列用ルーチン lnc2 と inv2　〃
IIIv : 小規模行列用ルーチン diag　〃

必要な記憶容量 [単位 MB; 1MB 以下は −, 1GB 以上は ∗]

n	idim(行列次元)	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
8	70	−	−	−	−	−	−	−	−
12	924	−	−	6.6	−	−	−	−	6.6
16	12870	−	2.7	∗	−	−	2.8	3.0	∗
20	184756	3.5	48	∗	4.9	7.8	49	51	∗
24	2704156	52	701	∗	72	113	836	877	∗
26	10400600	198	∗	∗	277	436	∗	∗	∗

S-820/80 上での CPU time[秒]
[1 秒以下は −, 記憶容量制限のため実行不可能なものは ∗]

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
8	−	−	−	−	−	−	−	−	−
12	−	−	−	2	−	−	−	−	2
16	−	−	−	∗	−	1	−	−	∗
20	5	10	1	∗	20	38	2	4	∗
24	80	225	∗	∗	448	839	∗	∗	∗
26	332	883	∗	∗	1758	2780	∗	∗	∗

VP-2600 上での CPU time[秒]

[1 秒以下は −, 記憶容量制限のため実行不可能なものは ∗]

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
8	−	−	−	−	−	−	−	−	−
12	−	−	−	2	−	−	−	−	2
16	−	−	−	∗	−	1	−	−	∗
20	6	8	2	∗	15	27	3	5	∗
24	108	155	∗	∗	306	565	∗	∗	∗

M-880/310 上での CPU time[秒]

[1 秒以下は −, 記憶容量制限のため実行不可能なものは ∗]

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
8	−	−	−	−	−	−	−	−	−
12	−	−	−	48	−	−	−	−	48
16	1	5	2	∗	11	21	4	11	∗
20	24	144	69	∗	290	503	131	283	∗

EWS-4800/220 上での CPU time[秒]

[1 秒以下は −, 記憶容量制限のため実行不可能なものは ∗]

n	sz	I	II	III	Iv	Iv′	IIv	IIv′	IIIv
8	−	−	−	−	−	−	−	−	−
12	−	1	1	816	2	4	2	3	816
16	5	34	21	∗	68	128	41	95	∗
20	104	852	∗	∗	1710	2988	∗	∗	∗

x2. 使用法

それぞれの行列規模のサブルーチンの使用例を示し，使用法を説明する．

2.1 大規模行列

大きな系に対して，行列要素を計算しながら Lanczos 法を使って 3 重対角化する．3 重対角化された行列の固有値はバイセクション法で求める．固有ベクトルは，固有値を求めた後もう一度 Lanczos 法のプロセスを繰り返して求めるか，あるいは逆反復法で計算する．逆反復法を使う場合には，逆反復の各ステップは CG 法で実行する (Appendix B 参照)．

本節のルーチン群においては行列要素を格納する必要がなく，その分だけ記憶容量は少なくて済む．固有値のみの場合には空間次元の長さを持つベクトル 2.5 本，固有ベクトルを nvec 本求めたいなら (2.5+nvec)£ 空間次元の記憶領域が必要である．本ルーチン群の短所としては行列要素を作る作業のため，次節の中規模用のルーチンに比べて処理速度が落ちる点が挙げられよう．速度と記憶容量についての詳しいデータは 1.6 節に掲げてある．

使用例 (固有値 4 つと基底固有ベクトルを求め，精度チェックをし，相関関数を計算する)

Sample program No. 1 (s1.f)

[1] スピン配列の生成

---

call sz(n, idim, szval, list1, list2) (szdy あるいは sztn は同等の機能でより高速)

[機能]

スピン数 n, S_z^{total}=szval で決定される空間上の許されるすべてのスピン配列を求める．スピン配列に番号を付け，その番号からスピン配列を与える表 (list1) と，スピン配列から番号を与える表 (list2) を作る．すべての規模の行列の計算に共通に使われるルーチンである．すべてのルーチンに先立って実行されなければならない．

[引数]

n: (入力; 整数)

スピン数．

idim: (入力; 整数)

n と szval で決定される空間の次元 (¸ 3)．上向きスピンの数を iu，下向きを id とすると iu+id=n, (iu ¡ id)/2=szval，これより iu=n/2 ¡ szval となり，行列次元は n から iu を選ぶ組み合わせの数 idim=$_nC_{iu}$ と計算される．例えば，n=20, szval = 0:0d0 だと，szval が 0 になるには上向きスピンの数が 10 だから，idim は 20 個のサイトから上向きスピンを置くサイトを 10 個選ぶ組み合わせの数 $_{20}C_{10}$ = 184756 である．

szval: (入力;8 バイト実数)

S_z^{total} の値．0.0d0 以上でなければならない．負の S_z^{total} の空間は，等価な正の空間に直して扱うこと．例えば，S_z^{total} = ¡ 1:0d0 は S_z^{total} = 1:0d0 の空間と同じ固有値を持ち，固有ベクトルは上向きスピンとした向きスピンを入れ替えれば同じになる．

list1(idim): (出力; 整数)

上向きスピンを 1, 下向きスピンを 0 としたとき list1(j) には今考えている空間内の j 番目のスピン配置が 10 進整数で入る．例えば，n=4, szval=0.0d0 のときは上向きスピン (1 で表わす) も下向きスピン (0) も共に 2 個ずつだから，idim=$_4C_2$=6 で，list1(1)=$(0011)_{2\text{進}}$=3, list1(2)=$(0101)_{2\text{進}}$ = 5;¢¢¢,list1(6)=$(1100)_{2\text{進}}$=12 となる．

list2(2, 0:2**15): (出力; 整数)

list1 の逆の表を与える．すなわち，10 進整数 i で表わされるスピン配置が今考えている空間内で j 番目のものだとすると，j=list2(1, ia)+list2(2, ib) の関係がある．ただし，ia は整数 i の下位のビットを切り出したもの [k = 2 ¤ ¤((n + 1)=2) として ia = IAND(i; k ¡ 1)], ib は同じく上位のビット [ib = IAND(i; IEOR(2 ¤ ¤n ¡ 1; k ¡ 1))=k] である．

これは小形と Lin の 2 次元サーチ法であり (4.2 節参照)，上位，下位の 2 つに分けて扱うことによって必要な配列次元が大幅に縮小されている．

[使用上の注意]

list1, list2 は以後の対角化ルーチンにそのまま渡されなければならない．sz においてはスピン配列を次々に生成して idim, n, szval との整合性を調べているので，ベクトル化率が低い．それゆえ，同じ n, szval, idim を持つ系をたくさん対角化したいときには，list1, list2 を一度作って外部ファイルに蓄え，読み込んで使用する方が速い．ただし，I/O 処理が遅いワークステーションなどでは必ずしもこの限りでない．

call lnc1(n, idim, ipair, bondwt, zrtio, ibond,
nvec, iv, E, itr, wk, ideclr, list1, list2)

[**機能**]

ハミルトニアンとスピン配置を与えて，基底状態を含む 4 個の固有値を求める．必要に応じて固有ベクトルを求める準備もする．

[**引数**]

n: (入力; 整数)

スピン数.

idim: (入力; 整数)

空間次元.

ipair(ibond*2): (入力; 整数)

格子の構造．ibond 個のボンドの両側のサイトの番号を 1 から n までの数の ibond*2 個の組として与える．例えば，図のような系の場合，data 文で

data ipair/1,2, 2,3, 3,4, 4,1, 1,3, 2,4/

とすればよい.

bondwt(ibond): (入力;8 バイト実数)

各ボンドの交換相互作用．ipair で決められたボンドの順に与える．例えば，上の例で $J_{12} = -1, J_{23} = -1.5, J_{34} = 0.3, J_{41} = -0.6, J_{13} = 0, J_{24} = 1$ ならば

data bondwt/−1.0d0, −1.5d0, 0.3d0, −0.6d0, 0.0d0, 1.0d0/

とする．ipair で与えたボンドの結合のデータの順番と一致する順に並べること．例えば上の data 文を

data bondwt/−1.0d0, −1.5d0, 0.3d0, −0.6d0, 1.0d0, 0.0d0/

とすると，$J_{13} = 1, J_{24} = 0$ という意味になる．

zrtio(ibond): (入力;8 バイト実数)

z 軸方向の異方性 Δ_{ij}．ipair で与えたボンドの結合のデータの順番と一致する順に並べる．

ibond: (入力; 整数)

ボンドの総数.

nvec: (入力; 整数)

あとで求める固有ベクトルの数．0 (固有ベクトル不要) から 4 の範囲でなければならない．lnc1 では lncv1 で固有ベクトルを求めるための準備データをルーチン内部で定義した common block/vecdat/に書き込むため，lncv1 で固有ベクトルを求める際には，必ずその前に nvec に 0 でない値を与えて lnc1 を実行しておかなくてはならない．

iv: (入力; 整数)

Lanczos 法の初期ベクトルで 1.0d0 と置く成分の番号．1 と idim の間の数．lnc1 では，初期ベクトルとして list1 で与えられるスピン配列のうち iv 番目の成分が 1.0d0 でその他の成分は 0.0d0 というものを選んでいる．番号の付け方は sz ルーチンで説明した通りのものである．ほとんどの場合，iv としてどのような値を与えても正しい固有値にほぼ一定の反復回数 (引数 itr に返ってくる) で収束する．ただし，iv=1 や iv=idim などは上向きスピンの配置が片寄った特殊なスピン配列なので，これらを初期条件とするとうまく収束しないことがある．1 と idim の中間の値を適当に選ぶと良い．下記の使用上の注意も参照のこと．

E(4): (出力;8 バイト実数)

基底状態を含めて 4 つの固有値が与えられる．有効数字は E(1) (基底エネルギー) と E(2)(第 1 励起エネルギーあるいは縮退があるときには基底エネルギー) については通常 10 桁程度はある．E(3), E(4) については系に依存するので，必要に応じて check1 ルーチンで確認すると良い．E(1), E(2) の詳細な信頼性をテストする場合も同様.

itr: (出力; 整数)

Lanczos 法の収束に要したステップ数．25 以上 5 刻みで与えられる．

wk(ideclr, 2): (作業領域;8 バイト実数）

ideclr: (入力; 整数)

作業領域の整合寸法．ideclr≥idim を満たす必要がある．

list1(idim): (入力; 整数)

sz ルーチンで求めたスピン配列.

list2(2, 0:215)**: (入力; 整数)

sz ルーチンで求めたスピン配列.

[**使用上の注意**]

収束のチェックは 25 ステップ目からはじめて 5 ステップごとに行ない，E(2) が 10^{-13} の相対精度で収束したときに処理を終了している．Lanczos 法の特性として，基底エネルギー E(1) は初期ベクトルが基底ベクトルに厳密に直交してない限り安定して求まるが，E(2) 以上は丸め誤差の影響によって縮退のない場合でも見かけ上の縮退 (E(2) と E(3) が同じ結果で返ってくるなど) を生ずることがある．収束の判定を E(2) で行なっているのは，E(3) あるいは E(4) にある程度以上の精度を要求して収束判定すると反復数が増し，丸め誤差の影響で見かけの縮退が生じ信頼性が損なわれる場合があるためである．したがって，E(3), E(4) の値を使用するときには check1 ルーチンで精度を確認することを薦める．E(1) と E(2) については見かけの縮退が生じる例はまれである．E(1)=E(2) となったときにはほとんど確実に真の縮退があるが，さらにそれを確認するには別の初期ベクトル (つまり別の iv) で lnc1 を実行し比較するとよい．さらに，固有ベクトルも複数の初期ベクトルから求めて直交化ルーチン orthg で縮退度を確認するという手も考えられる．なお，E(1) と E(2) が違う値でも基底状態に縮退がある場合もあるので注意が必要である．Appendix D の sample 10 を参照.

call lncv1(n, idim, ipair, bondwt, zrtio, ibond,
nvec, iv, x, itr, wk, ideclr, list1, list2)

[**機能**]

Lanczos 法によって固有ベクトルを求める．あらかじめ lnc1 ルーチンが nvec≠0 で実行され，固有ベクトルの計算に必要なデータが内部で定義された common block/vecdat/ に書き込まれている必要がある．

[**引数**]

x を除いては lnc1 と同じ．ただし，nvec, iv, itr は直前に実行された lnc1 での値と同じでなければならない．すなわち，lnc1 と lncv1 の間でこれらの変数を操作してはならない．また当然ながら，系を設定する変数 n, idim, ipair, bondwt, zrtio, ibond も lnc1 と同じでないといけない．

x(ideclr, nvec): (出力;8 バイト実数)

nvec($1 \leq$ nvec ≤ 4) 本の固有ベクトル．E(1) に対するものが x(1, 1) から x(idim, 1) まで, E(2) に対しては x(1, 2) から x(idim, 2) までという具合に返される．x の各要素の値の意味は，例えば基底波動関数については，スピン配列 list1(1) の振幅が x(1, 1), スピン配列 list1(2) の振幅が x(2, 1) などとなっている．sz ルーチンの list1 の項での例で言えば，

$$\psi_{\text{ground}} = \mathrm{x}(1,1) \times (0011)_{2\text{ 進}} + \mathrm{x}(2,1) \times (0101)_{2\text{ 進}} \cdots$$

である．励起状態についても同様．各波動関数は 1 に規格化されている．

[**使用上の注意**]

精度については，lnc1 での固有値に対するものと類似の注意が必要である．基底波動関数 (x(1, 1) から x(idim, 1) の値) は通常 6 桁程度以上の精度が期待できるが，励起波動関数の精度は高くない．精度の確認には check1 を用いる．特に高精度の波動関数が必要な場合には inv1 ルーチンを使う方がよい．

一般に物理量は適当な行列の波動関数での期待値であり，その精度は波動関数自体よりも高いことが多い．つまり，波動関数に存在する誤差が期待値を取る際にある程度打ち消されるのである．エネルギーについてはこの傾向が顕著で，ハミルトニアンの期待値を check1 で求めると波動関数の精度を大きく上回る結果が得られるのが普通である．これは波動関数の 2 次形式 $\langle\psi H\psi\rangle$ が $\psi = \psi_{\text{eigenstate}}$ で極値を取り，誤差に対して安定であるからである．その他の物理量についても波動関数の程度以上の精度が得られることがめずらしくない．

call inv1(n, idim, ipair, bondwt, zrtio, ibond,
Eig, iv, x, wk, ideclr, list1, list2)

[**機能**]

与えられた固有値の近似値に対して逆反復法によって固有ベクトルを求める．逆反復の各ステップは CG 法で実行する．lncv1 と違い，複数のベクトルを同時に計算することはできず，ただひとつの固有値 Eig に相当する固有ベクトルが返される．lncv1 と違い，他の方法で固有値 (近似値) がわかっていれば，事前に lnc1 を実行して内部変数の準備をする必要はない．

[**引数**]

Eig, iv, x, wk 以外の引数は lncv1 と同じ．

Eig: (入力;8 バイト実数)

固有値．この固有値に対する固有ベクトルが求められる．Eig は基底固有値，第 1 励起固有値等何でもよく，lnc1 で求まるものに限定されない．任意の固有値に対する固有状態が計算できる．与える固有値の精度が良いほど逆反復法の収束は速い．

iv: (入力; 整数)

逆反復の初期ベクトル．lnc1 での iv の説明を参照．lncv1 と違い，lnc1 と同じ iv を与える必要はない．

x(idim): (出力;8 バイト実数)

固有ベクトル．lncv1 と同じ並べ方で固有ベクトルの振幅が x(1) から x(idim) までに入る．

wk(ideclr, 4): (作業領域;8 バイト実数)

[**使用上の注意**]

inv1 で得られた固有ベクトルは通常 lncv1 で計算されたものより数桁以上高い精度を持つ．典型的には 8 ないし 10 桁程度の有効数字がある．check1 で確認すると良い．lncv1 と比較しての短所は，処理速度が数倍遅いこと (1.6 節の例を参照)，必要な作業領域が大きいこと (lncv1 では wk(ideclr, 2) だったものが wk(ideclr, 4) になっている)，また一度に 1 本のベクトルしか得られないことが挙げられる．必要に応じて lncv1 と inv1 を使い分けると良い．

[5] 固有値，固有ベクトルの精度チェック

call check1(n, idim, ipair, bondwt, zrtio, ibond, x, v, Hexpec, list1, list2)

[**機能**]

固有ベクトル x を与えて，それにハミルトニアンを掛けた結果を v に返す．また x でのハミルトニアンの期待値を Hexpec に返す．さらに，Hexpec を印字し，v と x の対応する成分の比 v(j)/x(j) も j=min(idim/3, 13) から idim まで max(1, idim/20) の間隔で印字する．この比は x が正しい固有ベクトルならば固有値になるはずであり，逆にこの比の固有値との一致の度合いを見ることによって固有ベクトルの精度を確かめられる．

[**引数**]

x, v, Hexpec 以外は lnc1 と同じ．

x(idim): (入力;8 バイト実数)
精度を調べる規格化された固有ベクトル．

v(idim): (出力;8 バイト実数)
入力ベクトル x にハミルトニアンを掛けて得られるベクトル．

Hexpec: (出力;8 バイト実数)
入力ベクトル x によるハミルトニアンの期待値．

[**使用上の注意**]

lnc1 で求めた固有値と Hexpec の比較によって固有値の精度がわかる．[3] で述べたように Hexpec のほうが lnc1 の与える値より精度が高い．いずれにしても，少なくとも Hexpec と lnc1 での計算値の一致するところまでは有効数字と言える．固有ベクトルは，v の各成分が x の対応する成分の Hexpec 倍になっているかどうかを調べることによって信頼性の限界がわかる．ここで注意しなければならないことは，もともと振幅が小さい成分 x(j) の値は有効数字の桁数が少なく，したがって v(j)/x(j) の Hexpec からのずれも大きくなるということである．物理量を求めるに当たっては，このような小さな振幅のスピン配列からの寄与は小さいので，有効数字の桁数の小ささは必ずしも障害にならないことに留意すべきである．例えば，x(10)=1.23456789d−10, x(20)=9.0d−20 のときには，x(20) が有効数字 2 桁であっても，ベクトル x を使って得られる物理量の有効数字は x(20) 以外が x(10) と同程度の有効数字を持つ限り，x(10) と同程度のものが期待できる．

[6]S_x, S_y の 2 点相関関数

call xcorr(n, idim, npair, nbond, x, sxx, list1, list2)

[**機能**]

固有ベクトル x を与えて，x, y 面内の 2 点相関関数を任意 (複数) のスピンペアについて求める．すべての規模の行列の計算に共通に用いられる．

[**引数**]

npair, nbond, x, sxx 以外は lnc1 と同じ．

npair(nbond*2): (入力; 整数)

相関関数を計算するスピンサイトのペアを入れる．入れ方は，lnc1 における ipair の与え方と同じ．

nbond: (入力; 整数)

相関関数を計算するスピンサイトのペアの数.

x(idim): (入力;8 バイト実数)

規格化された固有ベクトル.

sxx(nbond): (出力;8 バイト実数)

2 点相関関数．$\langle S^x_{\mathrm{npair}(1)} S^x_{\mathrm{npair}(2)} \rangle$ が sxx(1), $\langle S^x_{\mathrm{npair}(3)} S^x_{\mathrm{npair}(4)} \rangle$ が sxx(2), という具合に返される．$\langle S^y_i S^y_j \rangle$ は系の対称性より $\langle S^x_i S^x_j \rangle$ と等しい．

[**使用上の注意**]

npair で与えるペアは lnc1 等での ipair とは無関係である．1 から n までの任意のペアについての相関関数が求まる．

[7]S_z の 2 点相関関数

call zcorr(n, idim, npair, nbond, x, szz, list1)

[**機能**]

固有ベクトル x を与えて，z 方向の 2 点相関関数を任意 (複数) のスピンペアについて求める．すべての規模の行列の計算に共通に用いられる

[**引数**]

szz 以外は xcorr と同じ．

szz(nbond): (出力;8 バイト実数)

2 点相関関数．sxx と同様に $\langle S^z_{\mathrm{npair}(1)} S^z_{\mathrm{npair}(2)} \rangle$ が szz(1) に返される．szz(2) 以降も同様.

[**使用上の注意**]

xcorr に同じ．

call orthg(idim, ideclr, ev, norm, idgn, numvec)

[**機能**]

複数のベクトルを直交化し，線形独立性を調べ縮退度を求める．lncv1 や inv1 等でいくつかの異なる初期条件 iv から始めて得られたベクトルが互いに異なっているとき，そのうちいくつが線形独立であるかを見ることによってそのレベルの縮退度が決定できる．(本節最初の使用例では使っていない．Appendix D の sample 5, 10 を参照)．

[**引数**]

idim: (入力; 整数)
空間次元.

ideclr: (入力; 整数)
ev の整合寸法.

ev(ideclr, numvec): (入力および出力;8 バイト実数)
独立性を調べるベクトルを numvec 本入力する．直交化されたベクトルが返される．

norm(numvec): (出力; 整数)
直交化されたベクトルのノルム (1 か 0)．最初のベクトルのノルムが norm(1) に, 2 番目が norm(2)$\cdots$ に返される．

idgn:(出力; 整数)
縮退度．norm の成分中の 1 の数.

numvec:(入力; 整数)
調べるベクトルの数.

[**使用上の注意**]

直交化したあとのノルムが 10^{-15} 以下なら 0 ベクトルとみなして処理する．また直交化の作業中の丸め誤差の影響を評価するために，直交化後のベクトルどうしの内積を計算し，それが 10^{-10} 以下なら正しく直交化していると判定している．あまり多くのベクトルの直交化を試みると，誤差が積もって正しい結果が得られないことがある．またベクトルの精度が低いと正しい結果が得られないことがある．

2.2 中規模行列

あまり大きくない系に対して，行列要素のうち 0 でないものの位置と値をあらかじめ計算しておいてから，そのデータを使って Lanczos 法に基づいて 3 重対角化する．3 重対角行列の固有値はバイセクション法で求める．固有ベクトルは，固有値を求めた後もう一度 Lanczos 法のプロセスを繰り返して求めるか，あるいは逆反復法で計算する．逆反復法を使う場合には，逆反復の各ステップは CG 法で実行する．

行列要素を格納するための領域が必要であり，記憶容量は大規模用のルーチンに比べて相当大きなものが要求される．対角化を行う空間次元の長さを持つベクトルを系のボンドの数 £1:5 本蓄える程度の記憶容量が必要である．処理速度については，対角化の過程で行列要素を作る作業が不要のため，大規模用のルーチンに比べて大幅に高速化される．したがって記憶容量が許す限りは本節のルーチンを使用する方がよい．速度と記憶容量の例は 1.6 節に掲げてある．

サブルーチンのうち，大規模行列と共通に使われる sz, xcorr, zcorr, orthg については説明は省略する．

使用例 (固有値 4 つと基底固有ベクトルを求め，精度チェックをし，相関関数を計算する)

Sample program No. 6 (s6.f)

call elm2(n, idim, ipair, bondwt, zrtio, ibond,
elemnt, loc, ideclr, ic, list1, list2)

[**機能**]

ハミルトニアンとスピン配置を与えて，0 でない行列要素の位置と値を loc, elemnt にそれぞれ返す.

[**引数**]

n, idim, ipair, bondwt, zrtio, ibond, list1, list2 については 2.1 節の大規模行列の sz, lnc1 の項と同じ.

elemnt(ideclr, ic): (出力;8 バイト実数)

0 でない行列要素の値．第 j 行で k 番目にある 0 でない行列要素が elemnt(j, k) である．ただし，行内での順番は列の番号の順ではなく，ipair で指定したボンドの順になっている．つまりハミルトニアンの非対角項 (S_x, S_y の項) のうち，スピン ipair(1) とスピン ipair(2) にかかわる項 $-2J_{ij}(S^x_{\mathrm{ipair}(1)}S^x_{\mathrm{ipair}(2)} + S^y_{\mathrm{ipair}(1)}S^y_{\mathrm{ipair}(2)})$ がスピン配列 list1(j) に作用したとき生じる非対角項の値が elemnt(j, 1) に入る．elemnt(j, 2) は ipair(3), ipair(4) について，以後同様である．対角要素は elemnt(j, ic) に入る．

loc(ideclr, ic): (出力; 整数)

0 でない行列要素の位置．第 j 行で k 番目にある 0 でない行列要素の列番号が loc(j, k) である．すなわち第 j 行，第 loc(j, k) 列の行列要素が elemnt(j, k) である．

ideclr: (入力; 整数)

配列 elemnt, loc の整合寸法.

ic: (入力; 整数)

ボンド数 ibond に 1 を加えた数をいれる．1 行内の 0 でない行列要素の最大数である．

call lnc2(elemnt, loc, idim, ideclr, ic, nvec, iv, E, itr, wk)

[**機能**]

ハミルトニアンとスピン配置を与えて，基底状態を含む 4 個の固有値を求める．必要に応じて固有ベクトルを求める準備もする．事前に elm2 を実行して行列要素を求めておかねばならない．

[**引数**]

elemnt(ideclr, ic): (入力;8 バイト実数)

0 でない行列要素の値．elm2 で求めたもの．ルーチン終了後も同じ値が保たれる．

loc(ideclr, ic): (入力;8 バイト実数)

0 でない行列要素の位置．elm2 で求めたもの．ルーチン終了後も同じ値が保たれる．

idim: (入力; 整数)

空間次元．

ideclr: (入力; 整数)

elemnt, loc, wk の整合寸法．

ic: (入力; 整数)

ibond+1 を与える．1 行内の 0 でない行列要素の最大数．

nvec: (入力; 整数)

あとで lncv2 で求める固有ベクトルの数．0(固有値のみ) から 4 までの値でなければならない．lnc2 では lncv2 で固有ベクトルを求めるための準備データをルーチン内部で定義した common block/vecdat/に書き込むため，lncv2 で固有ベクトルを求める際には，必ずその前に lnc2 を nvec に 0 でない値を与えて実行しておかなくてはならない．

iv: (入力; 整数)

Lanczos 法の初期ベクトルで 1.0d0 と置く成分の番号．1 と idim の間の数．lnc2 では，初期ベクトルとして iv 番目の成分が 1.0d0 でその他の成分は 0.0d0 というものを選んでいる．lnc1 の iv の項の説明も参照せよ．

E(4): (出力;8 バイト実数)

基底状態を含めて 4 つの固有値が与えられる．有効数字は通常 E(1) (基底エネルギー) と E(2)(第 1 励起エネルギーあるいは縮退があるときには基底エネルギー) については 10 桁程度ある．E(3), E(4) については系に依存するので，必要に応じて check2 ルーチンで確認すると良い．E(1), E(2) の詳細な信頼性をテストする場合も同様．

itr: (出力; 整数)

Lanczos 法の収束に要したステップ数．25 以上 5 刻みで与えられる．

wk(ideclr, 2): (作業領域;8 バイト実数)

[**使用上の注意**]

lnc1 と同じ．

call lncv2(elemnt, loc, idim, ideclr, ic, nvec, iv, x, itr, wk)

[**機能**]

Lanczos 法によって固有ベクトルを求める．あらかじめ lnc2 ルーチンが nvec≠0 で実行され，固有ベクトルの計算に必要なデータが内部で定義された common block/vecdat/ に書き込まれている必要がある．

[**引数**]

x を除いては lnc2 と同じ．ただし，nvec, iv, itr は直前に実行された lnc2 での値と同じでなければならない．すなわち，lnc2 と lncv2 の間でこれらの変数を操作してはならない．また当然ながら，行列要素 elemnt, loc も lnc2 と同じでないといけない．

x(ideclr, nvec): (出力;8 バイト実数)

固有ベクトル．各波動関数は 1 に規格化されている．詳細は lncv1 の項を見よ．

[**使用上の注意**]

lncv1 に同じ．

call inv2(elemnt, loc, idim, ideclr, ic, iv, Eig, x, wk)

[**機能**]

与えられた固有値 (の近似値) に対して逆反復法によって固有ベクトルを求める．逆反復の各ステップは CG 法で実行する．lncv2 と違い，複数のベクトルを同時に計算することはできず，ただひとつの固有値 Eig に相当する固有ベクトルが返される．lncv2 と違い，他の方法で固有値 (近似値) がわかっていればそれを Eig に代入すればよく，事前に lnc2 を実行して内部変数の準備をする必要はない

[**引数**]

Eig, iv, x, wk 以外の引数は lncv2 と同じ．

Eig: (入力;8 バイト実数)

固有値．この固有値に対する固有ベクトルが求められる．Eig は基底固有値，第 1 励起固有値等 lnc2 で求まるものに限定されない．任意の固有値に対する固有状態が計算できる．固有値の精度が良いほど逆反復法の収束は速い．

iv: (入力; 整数)

逆反復の初期ベクトル．lnc1 での iv の説明を見よ．lncv2 と違い，lnc2 と同じ iv を与える必要はない．

x(idim): (出力;8 バイト実数)

規格化された固有ベクトル．lncv2 と同じ 並べ方で固有ベクトルの振幅が x(1) から x(idim) までに入る．

wk(ideclr, 4): (作業領域;8 バイト実数)

[**使用上の注意**]

inv1 に同じ．

call check2(elemnt, loc, idim, ideclr, ic, x, v, Hexpec)

[**機能**]
固有ベクトル x を与えて，それにハミルトニアンを掛けた結果を v に返す．また x でのハミルトニアンの期待値を Hexpec に返す．さらに，Hexpec を印字し，v と x の対応する成分の比 v(j)/x(j) も j=min(idim/3, 13) から idim まで max(1, idim/20) きざみで印字する．この比は x が正しい固有ベクトルならば固有値になるはずであり，逆に固有値との一致の度合いを見ることによって固有ベクトルの精度を確かめられる．

[**引数**]
x, v, Hexpec 以外は lnc2 と同じ．

x(idim): (入力;8 バイト実数)
精度を調べる規格化された固有ベクトル．

v(idim): (出力;8 バイト実数)
入力ベクトル x にハミルトニアンをかけて得られるベクトル．

Hexpec: (出力;8 バイト実数)
入力ベクトル x によるハミルトニアンの期待値．

[**使用上の注意**]
check1 に同じ．

2.3 小規模行列

小さな系に対して，行列要素をすべてあらかじめ求めておき，そのデータを使って House-holder 法に基づいて 3 重対角化する．行列次元は 3 以上でなければならない．3 重対角化行列の固有値はバイセクション法で求める．固有ベクトルは逆反復法で計算する．逆反復の各ステップは LU 分解による連立方程式の解法に従って実行する．

行列要素をすべて格納するための領域が必要であり，記憶容量は大規模用，中規模用のルーチンに比べて大きなものが要求される．対角化を行う空間次元 idim の 2 乗の大きさの配列を蓄える記憶容量がいる．数十次元以下の行列では Lanczos 法が安定して正しい結果を出さないことが多いため，このルーチンを用いる方がよい．逆に，約 100 次元以上になると，低い方の固有値を数個求めるという目的のためには Lanczos 法の方が良い．TITPACK Ver. 2 の小規模行列専用の部分はベクトルプロセッサを意識した並列計算のための最適化を特に行なってないので，大きな行列の固有値，固有ベクトルを上の方のレベルまでたくさん求めようとすると非常に時間がかかる可能性がある．速度と記憶容量についてのデータは 1.6 節に掲げてある．

サブルーチンのうち，大規模行列と共通に使われる sz, xcorr, zcorr, orthg については説明は省略する．

使用例 (固有値 4 つと基底固有ベクトルを求め，精度チェックをし，相関関数を計算する)

Sample program No. 11 (s11.f)

call elm3(n, idim, ipair, bondwt, zrtio, ibond, elemnt, ideclr, list1, list2)

[**機能**]
ハミルトニアンとスピン配置を与えて，行列要素を elemnt に返す.

[**引数**]
n, idim, ipair, bondwt, zrtio, ibond, list1, list2 については 2.1 大規模行列の sz, lnc1 の項と同じ.

elemnt(ideclr, idim): (出力;8 バイト実数)
行列要素．第 j 行第 k 列の行列要素が elemnt(j, k) である.

ideclr: (入力; 整数)
配列 elemnt の整合寸法.

call diag(elemnt, ideclr, idim, E, v, ne, nvec, eps, wk, iwk)

[**機能**]
行列要素を与えて，低い方から任意個数の固有値，固有ベクトルを求める．

[**引数**]
elemnt(ideclr, idim): (入力;8 バイト実数)
行列要素．elm3 で求めたものを入れる．diag 終了後は元の値を保っていない．

ideclr: (入力; 整数)
elemnt, vec, wk の整合寸法.

idim: (入力; 整数)
空間次元．2000 以下でなければならない．

E(ne): (出力;8 バイト実数)
低い方から ne 個の固有値が返される．

v(ideclr, nvec): (出力;8 バイト実数)
nvec 個の固有ベクトルが返される．E(1) に対応するものが v(1, 1) から v(idim, 1) まで，E(2) に対応するものが v(1, 2) から v(idim, 2) までという順に並べられる．各値の意味は lncv1 の項と同じ．

ne: (入力; 整数)
求める固有値の数．1 以上 idim 以下を指定する．

nvec: (入力; 整数)
求める固有ベクトルの数．0(固有ベクトル不要) から ne までの値を指定する．

eps: (入力;8 バイト実数)
許容相対誤差.

wk(ideclr, 8): (作業領域;8 バイト実数）

iwk(ideclr): (作業領域; 整数）

[3] 固有値，固有ベクトルの精度チェック

call check3(elemnt, idim, ideclr, x, v, Hexpec)

[**機能**]

固有ベクトル x を与えて，それにハミルトニアンをかけた結果を v に返す．また x でのハミルトニアンの期待値を Hexpec に返す．さらに，Hexpec を印字し，v と x の対応する成分の比 v(j)/x(j) も j=min(idim/3, 13) から idim まで max(1, idim/20) きざみで印字する．この比は x が正しい固有ベクトルならば固有値になるはずであり，逆に固有値との一致の度合いを見ることによって固有ベクトルの精度を確かめられる．

[**引数**]

x, v, Hexpec 以外は elm3, diag と同じ．x, v, Hexpec は check2 と同じ．

[**使用上の注意**]

diag を実行すると行列要素を入れる配列 elemnt の内容が書換えられるので，check3 を実行する前にもう一度 elm3 を呼ぶ必要がある．その他の注意点については check1 の項を参照.

§3. エラーメッセージ

エラーメッセージの種類とその原因および対策を述べる．Exx の形のメッセージでは処理がそこで終了しジョブは停止する．Wxx の形のメッセージの場合にはジョブは継続される．

E01: Variable szval given to sz out of range

[原因] スピン配列を生成するルーチン sz に与えられた szval の値が許容範囲外である．

[対策] szval を 0 から n/2−1 までの間で指定する．

E02: Incorrect idim or n given to sz

[原因] スピン配列を生成するルーチン sz に与えられた idim または n の値が許容範囲外である．

[対策] idim と n と szval の値が整合するよう指定する．2.1 節の sz の項を見よ．

E03: Incorrect data in ipair. Location : i j

[原因] 格子 (スピンペア) を指定する配列 ipair の要素の中に 0 以下か n+1 以上のものが見つかった．その位置は ipair(i) か ipair(j) である．

[対策] ipair に与えたデータをチェックする．特に個数 (ibond*2 本) が不足してないか調べる．

E04: ndim given to bisec exceeds 2000

[原因] diag で指定した次元数 idim が 2000 を超えていることがルーチン bisec でわかった．

[対策] diag では 2000 次元までしか扱えない．idim に 2000 以下を指定するか，lnc1, lnc2 を使う．

E05: ne given to bisec out of range

[原因] diag で指定した固有値の個数 ne が次元数 idim より大きいか 0 より小さいことがルーチン bisec でわかった．

[対策] ne を 0 と idim の間で指定する．

E06: Incorrect iv given to lnc1

[原因] lnc1 に与えられた初期条件 iv が 1 より小さいか idim より大きい．

[対策] iv を 1 から idim の間で指定する．

E07: Tridiagonalization unsuccessful in lnc1. Beta(i) is too small at i=xx.

[原因] lnc1 で Lanczos 法による 3 重対角化が，副対角要素が位置 xx で小さくなりすぎたため失敗した．

[対策] 初期条件 iv をかえて試みる．また数十次元以下の行列の場合にはルーチン diag を用いる．

E08: Incorrect iv given to lnc2

[原因] lnc2 に与えられた初期条件 iv が 1 より小さいか idim より大きい.

[対策] iv を 1 から idim の間で指定する.

E09: Tridiagonalization unsuccessful in lnc2. Beta(i) is too small at i=xx.

[原因] lncs で Lanczos 法による 3 重対角化が, 副対角要素が位置 xx で小さくなりすぎたため失敗した.

[対策] 初期条件 iv をかえて試みる. また数十次元以下の行列の場合にはルーチン diag を用いる.

E10: nvec given to diag out of range

[原因] diag に与えられた変数 nvec が 0 より小さいか ne より大きい.

[対策] nvec を 0 と ne の間で指定する.

W01: Wrong site number given to xcorr

[原因] xcorr に与えられたペアのサイト番号の中に 0 以下あるいは n+1 以上のものがある. あるいはペアを作るサイト番号の組が同一のものに指定されている. 処理をせずにメインルーチンに戻る.

[対策] サイト番号を 1 と n の間で正しい組を作るよう指定する.

W02: Wrong site number given to zcorr

[原因] zcorr に与えられたペアのサイト番号の中に 0 以下あるいは n+1 以上のものがある. あるいはペアを作るサイト番号の組が同一のものに指定されている. 処理をせずにメインルーチンに戻る.

[対策] サイト番号を 1 と n の間で正しい組を作るよう指定する.

W03: Number of vectors is less than 2 in orthg

[原因] orthg に与えられたベクトルの数 numvec が 1 以下である. 何もせずにメインルーチンに戻る.

[対策] 2 以上を指定する.

W04: Null vector given to orthg. Location is xx

[原因] orthg に与えられたベクトルのうち xx 番目のもののノルムが小さすぎる (10^{-20} 以下である). 直交化せずにメインルーチンに戻る.

[対策] そのベクトルを除いて直交化する.

W05: Non-orthogonal vectors at xx yy. Overlap : zz. Unsuccessful orthogonalization.

[原因] orthg で直交化が丸め誤差のためうまくいかなかった. 直交化試行後の xx 番目と yy 番目のベクトルの内積の絶対値が zz($\geq 10^{-10}$) である.

[対策] 誤差 zz を受け入れるかベクトルの本数を減らして試みる.

W06: Wrong value given to nvec in lnc1. Only the eigenvalues are calculated.

[原因] lnc1 に与えられた nvec が負であるか 4 を超えている．nvec=0 として処理を継続する．

[対策] nvec を 0 と 4 の間で指定する．

W07: lnc1 did not converge within 150 steps

[原因] lnc1 での Lanczos 法の反復が相対精度 10^{-13} では 150 ステップ以内に収束しなかった．近似的な固有値を返す．

[対策] 近似的な固有値を使って固有ベクトルを求め，check1 でさらに信頼性の高い固有値を計算する．あるいは小さい行列については diag を使う．または，ソースコード中で収束条件 (150 ステップ (100 番のループ)，10^{-13} の精度 (bisec を呼んだ後)) を緩めてみる．

W08: nvec given to lncv1 out of range

[原因] lncv1 に与えられた変数 nvec が 0 以下か 5 以上である．何もせずにメインルーチンに帰る．

[対策] nvec を 0 と 4 の間で指定する．

W09: Null vector given to check1

[原因] check1 に与えられたベクトルのノルムが 10^{-30} 以下である．何もせずにメインルーチンに戻る．

[対策] 間違って 0 ベクトルを与えたのでなく，ノルムの小さなベクトルを本当に調べたいのであれば，あらかじめ規格化しておいてから実行する．1 に規格化されたベクトルでないと Hexpec に正しい期待値は返らない．なお，lncv1, lncv2, inv1, inv2, diag で返されるベクトルはすべて規格化されている．

W10: Iterat in cg1 exceeds idim or 500. Approximate eigenvector returned. Iteration number in inv1 is xx.

[原因] inv1 で呼び出している cg1 ルーチン中の反復回数が idim か 500 を超えた．近似的な固有ベクトルを返す．inv1 での反復回数は xx である．

[対策] check1 で精度を確かめる．または初期条件 iv を変えて試みる．あるいは小さい行列なら diag を使う．

W11: inv1 did not converge

[原因] inv1 で逆反復法が 20 回以内で精度 10^{-12} 以内には収束しなかった．近似固有ベクトルを返す．

[対策] 精度の高い固有値を Eig に与える．あるいは lncv1 を用いる．

W12: Wrong value given to nvec in lnc2. Only the eigenvalues are calculated.

[原因] lnc2 に与えられた nvec が負であるか 4 を超えている．nvec=0 として処理を継続する．

[対策] nvec を 0 と 4 の間で指定する．

W13: lnc2 did not converge within 150 steps

[原因] lnc2 での Lanczos 法の反復が相対精度 10^{-13} では 150 ステップ以内に収束しなかった．近似的な固有値を返す.

[対策] 近似的な固有値を使って固有ベクトルを求め，check2 でさらに信頼性の高い固有値を計算する．あるいは小さい行列については diag を使う．または，ソースコード中で収束条件 (150 ステップ (100 番のループ)，10^{-13} の精度 (bisec を呼んだ後)) を緩めてみる．

W14: nvec given to lncv2 out of range

[原因] lncv2 に与えられた変数 nvec が 0 以下か 5 以上である．何もせずにメインルーチンに帰る．

[対策] nvec を 0 と 4 の間で指定する．

W15: Null vector given to check2

[原因] check2 に与えられたベクトルのノルムが 10^{-30} 以下である．何もせずにメインルーチンに戻る．

[対策] 間違って 0 ベクトルを与えたのでなく，ノルムの小さなベクトルを本当に調べたいのであれば，あらかじめ規格化しておいてから実行する．1 に規格化されたベクトルでないと Hexpec に正しい期待値は返らない．なお，lncv1, lncv2, inv1, inv2, diag で返されるベクトルはすべて規格化されている．

W16: itr in cg2 exceeds idim or 500. Approximate eigenvector returned. Iteration number in inv2 is xx.

[原因] inv2 で呼び出している cg2 ルーチン中の反復回数が idim か 500 を超えた．近似的な固有ベクトルを返す．inv2 での反復回数は xx である．

[対策] check2 で精度を確かめる．または初期条件 iv を変えて試みる．あるいは小さい行列なら diag を使う．

W17: inv2 iteration did not converge

[原因] inv2 で逆反復法が 20 回以内で精度 10^{-12} 以内には収束しなかった．近似固有ベクトルを返す.

[対策] 精度の高い固有値を Eig に与える．あるいは lncv2 を用いる．

W18: Null vector given to check3

[原因] check3 に与えられたベクトルのノルムが 10^{-30} 以下である．何もせずにメインルーチンに戻る．

[対策] 間違って 0 ベクトルを与えたのでなく，ノルムの小さなベクトルを本当に調べたいのであれば，あらかじめ規格化しておいてから実行する．1 に規格化されたベクトルでないと Hexpec に正しい期待値は返らない．なお，lncv1, lncv2, inv1, inv2, diag で返されるベクトルはすべて規格化されている．

TITPACK Ver. 2 ではアルゴリズムとして特に新しいものを用いているわけではない．Lanczos 法，逆反復法など以前からよく知られているものばかりである．ごく簡単な説明は Appendix B にしてあるが，詳細については参考文献を参照してほしい．これらのアルゴリズムを使って実際に TITPACK Ver. 2 をコーディングするに際していくつか注意した点がある．それを以下で解説する．

4.1 ビット演算

ハミルトニアン行列をベクトルに掛けるという演算が Lanczos 法のような反復法の系統のアルゴリズムには頻繁にでてくる．TITPACK Ver. 2 での大規模行列の処理のようにハミルトニアンが行列の形であらわに与えられてないときにこの掛け算をプログラム上で実現するには，2 つの整数の対応するビットどうしの論理演算を利用する．ルーチン mltply に典型的に現れているコーディングなので参照しながら読むとよい．

まず対角要素 (ハミルトニアンの S_z 成分の部分) を考える．第 k 番目のスピンペア ipair(k*2−1) ≡i1 と ipair(k*2)≡i2 に相当する対角要素を H_k としよう．

$$H_k = -2J_{\mathrm{i_1 i_2}}\Delta_{\mathrm{i_1 i_2}}S^z_{\mathrm{i_1}}S^z_{\mathrm{i_2}}$$

これにベクトル v の第 j 成分 v(j) が掛かるとして，i1 のスピン状態と i2 の状態が平行なら結果は $-0.5J_{\mathrm{i1,i2}}\Delta_{\mathrm{i1,i2}}$v(j) であり，反平行なら逆符号となる．j はスピン状態の番号を表わしており，2.1 節の sz の項で説明したように，list1(j) を 2 進法で表わしたものが実際のスピン状態となる．i1 に相当するビット (2 進表現で右から isite1≡i1−1 番目) と i2 に相当する isite2≡i2−1 番目のビット両方に 1 を立てたもの is($=1\times 2^{\mathrm{isite1}}+1\times 2^{\mathrm{isite2}}$) を作り，それを使って論理積を取る関数 IAND で list1(j) から 2 つのビットを切り出す．もし 2 つのビットの状態が同じ (平行スピン) なら，切り出した結果 ibit は 0(2 つとも下向き，つまり $0\times 2^{\mathrm{isite1}}+0\times 2^{\mathrm{isite2}}$) か is(2 つとも上向き) となるし，反平行ならその他の値を取る．だから ibit が is か 0 に等しいかどうかで処理を分ければよい．

非対角要素についても同様である．2 つのスピンが平行ならば非対角要素は作用せず，何もする必要がない．上記の方法で反平行であることがわかったなら，反平行スピンどうしを反転して新しいスピン状態を作る．反転するには，関数 IEOR(排他的論理和) を使う．これは，2 つの整数の対応するビットどうしが (1, 0) または (0, 1) ならそのビットに 1 を返し，(1, 1) または (0, 0) なら 0 を返す．つまり，(j1, j2) の組で j1=1 なら返ってくるのは j2 を反転したものであり，j1=0 なら j2 そのものが返ってくる．したがって，list1(j) の 2 つのビット isite1 と isite2 を反転するには，これらのビットだけに 1 を立て，その他のビットは 0 とした is と list1(j) の IEOR を取るとよい．こうして得られた新しいスピン配列 iexchg がもとのスピン配列表中で何番目にあるかを次節で述べる 2 次元サーチ法で

調べて，その番号のところのベクトル v の成分を積和を記録するベクトル v0 の第 j 成分に足し上げる．

4.2 2 次元サーチ法

スピン配列が整数として与えられたときに，それが何番目の配列かを知る必要が生じることがある．単純に考えると，スピン配列 i を与えればその番号を返す配列 list2′ をあらかじめ作っておけば一度の参照 list2′(i) だけで番号がわかり何の問題もないように思える．実際 TITPACK Ver.1 ではそうしていた．しかし，スピン数 n が増加してくると，スピン配列 i として許される整数の値の最大値が $2^n - 1$(つまり 2 進表現で n 個の 1 が並んでいる) と指数関数的に増大するため，list2′ も非常に大きなものになり記憶容量の点で困難が生じる．

一般に，$S_z^{\rm total}$ が一定の空間では $2^n - 1$ 個のスピン配列がすべて必要なわけではない．例えば，n=20, $S_z^{\rm total} = 0$ の場合，$2^n - 1 = 1048575$ であるが，実際に必要な配列は 20 個のスピンサイトから上向きスピンを配置する 10 のサイトを選ぶ組み合わせの数 ${}_{20}{\rm C}_{10} = 184756$ である．したがってその差 1048575−184756=863819 個の分の記憶領域は list2′ の中で無駄になっている．この不必要な領域を使わないようにすると同時に，list2′(i) の引数として許される最大数 $2^n - 1$ を引き下げることが出来れば大幅な記憶容量の節約になり，より大きな系が取り扱えるようになる．

小形正男 (私信) と H.Q. Lin(参考文献 1) は，スピン配列を表わす数 j を 2 進表現で 2 つの部分に分けて操作することによってこの問題を解決した．簡単な例で説明しよう．スピン数 n=6，$S_z^{\rm total}$=szval=0.0d0 とする．上向き，下向きいずれも 3 個のスピンがある．スピン配列に i の大きさの順で番号を付け，i の 2 進表現を下 3 桁 (ia) と上 3 桁 (ib) に分ける．表には最初の 6 個のスピン配列を例示してある．

番号	スピン配列 i(2 進)	ib	ia	jb	ja	ja+jb
1	000111	000	111	0	1	1
2	001011	001	011	1	1	2
3	001101	001	101	1	2	3
4	001110	001	110	1	3	4
5	010011	010	011	4	1	5
6	010101	010	101	4	2	6

そして下の桁のスピン配列 ia と上の桁のスピン配列 ib に次の規則で別々に番号 ja, jb を付ける．まず ia と ja の対応を述べる．jb が一定値の間 (つまり ib が一定の間) は ja は 1 から始まって 1 ずつ増える．ib が変わり，jb もそれにしたがって更新されると，ja はまた 1 からやり直す．こうすると各 ia に対して ja が一意的に決まる．(異なる ia が同じ ja

に対応することもあるが構わない．) 次に jb は最初は 0 とし，ib が更新されるとひとつ前の ja+jb を新たな jb とする．以上の操作でスピン配列 i が与えられたとき，その下の桁 ia と上の桁 ib の番号 ja, jb が決まる．そして，それらの和 ja+jb が常に全体のスピン配列番号と一致する．

記憶するべき量は，下 3 桁の配列 ia が何番目か (ja) を与えるリストと上 3 桁についても同様のリストである．完全な 6 桁のスピン配列の番号を一ぺんに取り扱おうとすると 2^6-1 の大きさの領域 list2′ を用意しないといけないが，3 桁ずつに分けると 2^3-1 のを 2 本用意すればよい．n=6 ぐらいではほとんど差がないが，例えば n=26 だとおよそ 2^{26} と 2×2^{13} となって，前者はとても主記憶に入らない大きさだが，後者だと全く問題ない．TITPACK Ver. 2 上ではスピン数 32 以上を取り扱うことはビット表現上出来ないので，最大 n=31 として，およそ 2^{15} の大きさの配列 list2 を 2 本 list2$(2,0:2**15)$ 用意してある．

与えられた整数 iexchg の 2 進表現を下の桁と上の桁に分けるには次のようにする．まず下の桁については，下の桁の部分に 1 ばかりを立てた整数 irght を用意してそれと iexchg の IAND をとる．上の桁については上の桁に相当するビットに 1 を立てた ilft との IAND をとり，真ん中のビットだけに 1 を立てた ihfbit で割ることによって 2 進表現を右端に寄せるだけでよい．

4.3 2 つのベクトルだけを使った Lanczos 法

Lanczos 法を実行するには，普通 3 本のベクトルを記憶する必要がある．現在のベクトル $\mathbf{v}_1$, ひとつ前のもの $\mathbf{v}_0$，$\mathbf{v}_1$ にハミルトニアンを掛けて得られるベクトル $\mathbf{v}_2$ の 3 つを蓄えておき適当な係数を掛けて和を取る操作が行なわれる (Appendix B)．松下操 (私信) は，$\mathbf{v}_2$ は $\mathbf{v}_0$ とのある線形結合の形でのみ必要になることに着目して，ベクトル 2 本で Lanczos 法が実行できることを指摘した．3 本が 2 本になるだけのように思えるかもしれないが，主記憶容量ぎりぎりの計算をするときにはこれが大きな違いになってくる．

TITPACK Ver. 2 中のルーチン mltply において，入力ベクトル v_0 は Appendix B の (3) 式中 $-\beta_{i-1}\mathbf{v}_{i-1}$ に相当し，mltply で返される v_0 は $H\mathbf{v}_i-\beta_{i-1}\mathbf{v}_{i-1}$ になっている．mltply では同時に $\langle \mathbf{v}_i H\mathbf{v}_i\rangle=\alpha_i$ も求めている．

§5. §2 を読まずに使おうとしているあなたへ

Appendix D には状況に応じてパラメータを変えればそのまま走るサンプル・プログラムを載せてある．次の手順にしたがって書換えれば使えるようになっている．精度やパラメータの条件等で注意しないといけない点を §2 に記述してあるので，使用するルーチンの部分は読んでおくことをおすすめする．

1. 解きたい問題に応じてハミルトニアン

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

   に出てくるパラメータ J_{ij}, Δ_{ij} それに格子形 ($\langle i,j \rangle$ は何番目のサイトどうしのペアか) を決める．サイトの番号は 1 から n(スピン数) の間で連続して付けること．また全スピンの z 成分 $S_z^{\rm total}$ がどういう値の空間を調べるかを決める．$S_z^{\rm total}$ が負の空間は扱えない．

2. 行列次元 idim を求める．これは n と $S_z^{\rm total}$ から決まる．上向きスピンの数を iu，下向きを id とすると iu+id=n, (iu−id)/2=$S_z^{\rm total}$，これより iu=n/2−$S_z^{\rm total}$ となり，行列次元は n から iu を選ぶ組み合わせの数 idim=${}_{\rm n}{\rm C}_{\rm iu}$ と計算される．

3. idim が 100 以下なら小規模用ルーチンを使う．Sample 11 である．書換えるところは次のとおり．

   a. parameter 文の n, idim, ibond(ボンドの総数).

   b. dimension 文の npair(2), sxx(1), szz(1)．相関関数を k 組のスピンペアについて求めたいなら 2 を 2*k にし，1 を k にする．それに対応して call xcorr の上の npair(1)=1, npair(2)=2 にスピンペアを入れる．例えば，スピンペア 2, 5 と 4, 7 について計算したいなら，npair(1)=2, npair(2)=5, npair(3)=4, npair(4)=7 とする．さらに xcorr, zcorr の引数の 1 を k とする．

   c. 3 つのデータ文．bondwt には J_{ij} の値を ibond 個入れる．Sample 11 はすべて −1 (反強磁性) の場合である．zrtio には Δ_{ij} の値を ibond 個入れる．Sample 11 はすべて 1 (等方的ハイゼンベルクモデル) の場合である．ipair には格子形を与える．どのサイトどうしがつながっているかを ibond 組 (ibond*2 個) 指定する．Sample 11 は 1 次元で周期境界の例である．

   d. call sz での 3 つめの引数 0.0d0 のところに $S_z^{\rm total}$ の値を入れる．

   e. 固有値の数を指定する ne=4 と固有ベクトルの数を指定する nvec=1．nvec を 2 以上にした場合には，dimension 文の v(idim) を v(idim, nvec の値) とすること．また精度チェックや相関関数の計算を基底固有ベクトル以外について行ないたい場合には，check3, xcorr, zcorr の引数 v を v(1, ベクトルの番号) とする．ベクトルの番号は基底ベクトルを 1，第 1 励起ベクトルを 2 という順で数える．

4. idim が 100 以上のとき，大規模用を使うか中規模用を使うかを決める．中規模用ルーチンでの所要記憶容量のおよその値は次の式で与えられる．

$$\text{idim} \times \text{ibond}(\text{ボンドの総数}) \times 1.5 \times 8/1024^2 \quad (\text{MB})$$

これが，使っている計算機の主記憶に入るならば中規模用を使った方が処理速度の点で有利である．

5. 大規模用を使って固有値と基底固有ベクトルを求めるとき．

Sample 1 か Sample 2 を使う．特に高精度の固有ベクトルがほしいときには Sample 2 を，そうでなければ Sample 1 にする．それぞれで書換えるべき点は上記 3. と同じであるから参照のこと．

6. 大規模用を使って固有値 4 つと固有ベクトルを nvec 本求めるとき．

Sample 3 を使う．書換えは，上記 3 に加えて parameter 文の nvec も自分の必要な値に変える．Sample 3 では最後のベクトル (nvec 番目) の精度チェックをしている．2.1 節の lncv1 と inv1 の項で述べたように，一般に励起状態のベクトルの精度は lncv1 で得られたものについてはあまり高くない．

7. 励起固有ベクトルが十分な精度でほしいとき．

Sample 4 を使う．書換えは，上記 3 に加えて，inv1 の引数 E(3) を必要な励起レベルで置き換える．

8. 縮退度を調べたいとき．

Sample 5 にしたがって orthg を呼ぶ．Sample 5 では初期ベクトルを変えて 2 本固有ベクトルを求め，それらを v に蓄えてから orthg に渡し，idgn に返される値として縮退度を求めている．問題に応じて変更するべき点は，上記 3 の他に，予想される縮退度 (idg とする) を dimension 文の v(idim, 2), norm(2) の 2 の所に入れることと 10 番のループの idim/2 を idim/idg としさらに orthg の最後の引数も idg とすることである．

9. 中規模用ルーチンが使える場合．

Sample 6 から 10 を利用する．大規模用の Sample 1 が中規模用の Sample 6 に，以下同様に Sample j が Sample (j+5) に対応する．問題に応じて書換えるべき所は大規模用の対応するものと同じである．ただし，Sample 10 においては Sample 5 と違って縮退度の可能な値を念のため 3 まで取っているので上記 8 で 2 とした部分を 3 と読み替える．

§6. 入手法など

TITPACK Ver. 2 のソースコードはこのマニュアルの最後にある．また Ver. 1 と同様に，東大大型計算機センター (HITAC システム) のデータセット A85085.TITPACK2.FORT にも入っており，すべてのユーザに読み出しを許可してある．大規模用，中規模用，小規模用，共通ルーチンごとに別々のメンバー large, medium, small, common になっている．また Appendix D のサンプル・プログラムがメンバー sample1 から sample11 に入っている．随時お知らせを載せるメンバー readme も作成する予定である．

またプログラムの内容についての問い合わせには出来るだけ誠意を持って回答するが，コンパイルの仕方やファイルの東大から自分のセンターへの転送の方法など計算機システムの利用の仕方に関する問題は自分で処理してほしい．

大型計算機センターでの配布は一般的ではなくなった。ホームページでの配布に変更し、東大センターのファイルの維持管理と公開は中止した。(1999 年 7 月 21 日)。

`http://www.stat.phys.titech.ac.jp/~nishi/`

参考文献

1. H.Q. Lin, Phys. Rev. B**42** (1990) 6561.
2. 森正武, FORTRAN77 数値計算プログラミング (岩波書店) 1988.
3. 戸川隼人, マトリクスの数値計算 (オーム社) 1971.
4. E.R. Gagliano, E. Dagotto, A. Moreo and F.C. Alcaraz, Phys. Rev. B**34** (1986) 1677.

Appendix A. ユーザは直接使わない下位ルーチンの説明

ソースコードを読んで自分の目的に合わせてさらに最適化しようという人のために，下位ルーチンの機能を説明する．

[1]Lanczos 法による 3 重対角化の実行 (固有値計算)

call lnc1z(n, idim, ipair, bondwt, zrtio, ibond,
nvec, iv, E, itr, v1, v0, list1, list2)

[**機能**]

lnc1 はパラメータをチェックするだけで実際の 3 重対角化の作業はすべてこの lnc1z で行う．lnc1z での作業領域はベクトル 2 本 (v0, v1) であるが，lnc1 のユーザは 1 本の 2 次元配列 wk を呼べば良く，プログラミングの負担が軽減されるため，このような形にした (参考文献 2)．もっとも，作業領域を引数に並べる手間が減っても，整合寸法 ideclr を宣言しないといけなくなるために結局引数の数は lnc1 と lnc1z で変わってないが，inv1 などでは明らかにユーザの負担が軽減される．それらとのコーディングの一貫性を保つため，lnc1 についても敢えて実際の作業は別のルーチンに渡すようにした．

[**引数**]

作業領域を表す引数 v1, v0 が wk, ideclr に取って変わったこと以外は lnc1 と同じ．v1(idim), v0(idim) はいずれも 8 バイト実数.

[2]Lanczos 法による 3 重対角化の実行 (固有ベクトル計算)

call lncv1z(n, idim, ipair, bondwt, zrtio, ibond,
nvec, iv, x, ideclr, itr, v1, v0, list1, list2)

[**機能**]

lnc1z と同様に，lncv1 から作業領域を引き継いで，固有ベクトルの計算をする．

[**引数**]

作業領域を表す引数 v1, v0 が wk, ideclr に取って変わったこと以外は lncv1 と同じ．v1(idim), v0(idim) はいずれも 8 バイト実数.

[3]Lanczos 法の実行に必要な行列の積和と期待値の計算

call mltply(n, idim, ipair, bondwt, zrtio, ibond, v1, v0, prdct, list1, list2)

[**機能**]

2 つのベクトル v1, v0 の入力に対して $H \times \mathrm{v1} + \mathrm{v0}$ を返す．また v1 でのハミルトニアン H の期待値も返す.

[**引数**]

v0, v1, prdct 以外は lnc1 と同じ.

v1: (入力;8 バイト実数)

入力データベクトル

v0: (入力および出力;8 バイト実数)

入力データベクトル v1 と v0 に対して，$H \times \mathrm{v1} + \mathrm{v0}$ を求めて v0 に入れて返す.

prdct: (出力;8 バイト実数)

v1 でのハミルトニアンの期待値．v1 が規格化されていることを前提としてる．規格化されてないと v1 のノルム $\langle(\mathrm{v1})^{\mathrm{T}}\mathrm{v1}\rangle$ が掛かった値を返す.

[4] 逆反復法の実行

---

call inv1z(n, idim, ipair, bondwt, zrtio, ibond,
Eig, iv, x, b, p, r, y, list1, list2)

[**機能**] 逆反復法を実行する．lnc1z と同じように，作業領域を表す引数の数を減らすために inv1 から inv1z を呼び出すようにしている.

[**引数**]

b, p, r, y が wk, ideclr に取って変わった以外は inv1 と同じ.

b(idim), p(idim), r(idim), y(idim): (作業領域;8 バイト実数)

[5]CG 法による連立方程式の解

---

call cg1(n, idim, ipair, bondwt, zrtio, ibond,
Eig, x, b, p, r, y, itr, list1, list2)

[**機能**] 逆反復法で新しい近似固有ベクトル ψ_i を求める作業 $(H - \mathrm{Eig})^{-1}\psi_{i-1} = \psi_i$ を連立方程式 $(H - \mathrm{Eig})\psi_i = \psi_{i-1}$ を解くことによって実行する.

[**引数**]

inv1 との違いは inv1 の iv, x, wk, ideclr の部分が x, b, p, r, y, itr となっただけ．残りの引数の意味は同じ.

x(idim): (出力;8 バイト実数)

新しい近似固有ベクトル．上記の機能の説明で言うと ψ_i.

b(idim), p(idim), r(idim), y(idim): (作業領域;8 バイト実数)

itr: (出力; 整数)

収束までに要したステップ数が 5 の倍数で返される．収束条件は，CG 法で残差のノルムが方程式の右辺 (ψ_{i-1}) のノルムの 1.0×10^{-9} 倍以下になることとしてある.

[**使用上の注意**]
反復回数は 500 と idim の小さい方を最大値とする．経験的に, これ以上の反復を試みても無駄であり，初期条件 iv を変えて inv1 を呼ぶか別のルーチンを使用する方が良い．

[6] 中規模行列における対角化の実行

中規模行列における対角化の実行ルーチン群 lnc2z, lncv2z, inv2z, cg2 が主要ルーチン群 lnc2, lncv2, inv2 に対して受け持つ役割は，大規模行列における対応する関係 (lnc1z と lnc1 など) と同じである．呼出しの形式だけを列挙しておく．

call lnc2z(elemnt, loc, idim, ideclr, ic, nvec, iv, E, itr, v1, v0)

v1(idim), v0(idim) はいずれも 8 バイト実数.

call lncv2z(elemnt, loc, idim, ideclr, ic, nvec, iv, x, itr, v1, v0)

v1(idim), v0(idim) はいずれも 8 バイト実数.

call inv2z(elemnt, loc, idim, ideclr, ic, iv, Eig, x, b, p, r, y)

b(idim), p(idim), r(idim), r(idim) はいずれも 8 バイト実数.

call cg2(elemnt, loc, idim, ideclr, ic, Eig, x, b, p, r, y, itr)

x, b, p, r, y については上記 cg1 の項を見よ．その他は inv2z と同じ．

[7]Householder 法による 3 重対角化

call hshldr(elemnt, ideclr, idim, alpha, beta, c, w, p, q)

[**機能**] 行列要素がすべて与えられたときにその行列を Householder 法で 3 重対角化する．コーディングの主要部分は参考文献 2 による．

[**引数**]
elemnt, ideclr, idim はルーチン diag と同じ．

alpha(idim): (出力;8 バイト実数)
対角要素.

beta(idim): (出力;8 バイト実数)
副対角要素.

c(idim): (出力;8 バイト実数)
固有ベクトルを求める際に必要になる規格化因子.

w(idim), p(idim), q(idim): (作業領域;8 バイト実数)

[8]3 重対角行列の固有ベクトル (3 重対角化される前の基底での表示)

call vec3(E, elemnt, ideclr, idim, ne, nvec,
di, bl, bu, bv, cm, lex, alpha, beta, c, v)

[**機能**] 3 重対角行列の固有ベクトルを逆反復法で求め，3 重対角化する前の基底での表示に変換して返す．逆反復法の各ステップにおける連立方程式は，行列を LU 分解し前進代入，後退代入して解く．コーディングの主要部分は参考文献 2 による．

[**引数**]

elemnt, ideclr, idim はルーチン diag と同じ．

E(ne): (入力;8 バイト実数)
固有値．bisec で求めた近似値を入れる．

ne: (入力; 整数)
入力する固有値の数．idim 以下．

nvec: (入力; 整数)
求める固有ベクトルの数．1 以上 ne 以下．

di(idim), bl(idim), bu(idim), bv(idim), cm(idim): (作業領域;8 バイト実数)

lex(idim): (作業領域; 整数)

alpha(idim): (入力;8 バイト実数)
対角要素．

beta(idim): (入力;8 バイト実数)
副対角要素．

c(idim): (入力;8 バイト実数)
3 重対角化する前の基底での表現にもどすために必要な規格化因子．hshldr で求めたもの．

v(ideclr, nvec): (出力;8 バイト実数)
固有ベクトル．E(1) に対応するものが v(1, 1),···,v(idim, 1) に，E(2) に対応するものが v(1, 2),···,v(idim, 2) にという順序で返される．3 重対角化する前の基底 (sz で生成したもの) での表示である．

[9] サイトデータのエラーチェック

call datack(ipair, ibond, n)

[**機能**]

ipair に与えたサイトのペアのデータの中に 0 以下か n+1 以上の数がないかどうか調べる．そのようなデータが見つかるとエラーメッセージを出して停止する．

[**引数**] どの引数も lnc1 における同名のものと同じ意味を持つ．

[10] バイセクション法による 3 重対角行列の固有値の計算

call bisec(alpha, beta, ndim, E, ne, eps)

[**機能**]

3 重対角行列を与えて，バイセクション法によりその固有値を低いレベルから ne 個求める．コーディングの主要部分は参考文献 2 による．

[**引数**]

alpha(ndim): (入力;8 バイト実数) 3 重対角行列の対角要素.

beta(ndim): (入力;8 バイト実数) 3 重対角行列の副対角要素． 最後 (第 ndim 番目) の要素の値は任意.

ndim: (入力; 整数) 3 重対角行列の次元.

E(ne): (出力;8 バイト実数) 低い方から ne 個の固有値が返される．E(1) が基底レベル，E(2) が第 1 励起レベルという順.

eps: (入力;8 バイト実数) 許容相対誤差.

[11]3 重対角行列の固有ベクトル (3 重対角化されたときの基底での表示)

call vec12(E, ndim, nvec, di, bl, bu, bv, cm, lex)

[**機能**] 3 重対角行列の固有ベクトルを逆反復法で求める．逆反復法の各ステップにおける連立方程式は，行列を LU 分解し前進代入，後退代入して解く．vec3 と違い，3 重対角化する前の基底での表示に変換することはしない．結果は内部で定義してある common block/vecdat/に書き込まれ，後で lncv1, lncv2 によって元の表示での固有ベクトルを求める際に利用される．3 重対角行列の対角要素，副対角要素 alpha, beta は lnc1, lnc2 から common block/vecdat/で渡される．コーディングの主要部分は参考文献 2 による．

[**引数**]

nvec, di, bl, bu, bv, cm, lex はルーチン vec3 と同じ．

E(4): (入力;8 バイト実数)

基底固有値およびその上の 3 つの固有値．bisec で求めた近似値を入れる．

ndim: (入力; 整数) 3 重対角行列の次元.

Appendix B. Lanczos 法，逆反復法，CG 法

大規模行列と中規模行列の処理で採用した主なアルゴリズムである Lanczos 法，逆反復法および CG 法について参考文献 3 に基づいて簡単に解説する．より詳しい説明については参考文献 3 を参照してほしい．また小規模行列の 3 重対角化に使う Householder 法や 3 重対角行列の対角化のためのバイセクション法，LU 分解等については参考文献 2 に明快な説明があり，プログラム例も載っているのでそちらを参照されたい．TITPACK Ver. 2 では Householder 法，バイセクション法，LU 分解による連立方程式の解法の部分は文献 2 のプログラム例を適宜書換えて使用している．

B.1 Lanczos 法

Lanczos 法にはいろいろな側面があるが，大規模行列を効率良く対角化するという目的から見ると，反復法の改良として Lanczos 法を考えるのがわかりやすい．反復法とは，行列 H の絶対値最大の固有値を求めるのに，適当な初期ベクトル $\mathbf{v}_0$ から始めて，繰り返し H を掛けていくと絶対値最大の固有値を持つ固有ベクトルに収束するというものである．つまり m 次元の行列 H の固有値，固有ベクトルを $E_j, \psi_j (j = 1, \cdots, m)$ とするとき，初期ベクトル $\mathbf{v}_0$ を固有ベクトルで展開すると

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

である．これに H を k 回掛けると

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

となり，k の増加につれて指数関数的に絶対値最大の固有値を持つ固有状態の相対的な重みが j についての和の中で増加するのである．

単純な反復法の収束を加速する方法として，$\mathbf{v}_k$ から $\mathbf{v}_{k-1}$ 以前のベクトルの成分を引くことによって，任意に選んだ初期条件の影響を出来るだけ速く脱出して正しい固有ベクトルに到達するという操作が考えられる．この，前のベクトルの成分を差し引くという操作が実は 3 重対角化するのと同じことになっているのである．これを見てみよう．

3 重対角化されたあとの行列を T とし，そのための変換行列を V とすると，$T = V^{-1}HV$，すなわち $VT = HV$ となる．V の列ベクトルを $\mathbf{v}_1, \mathbf{v}_2, \cdots$ とし，T の対角要素を $\alpha_1, \alpha_2, \cdots$，副対角要素を $\beta_1, \beta_2, \cdots$，とすると $VT = HV$ という関係は

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

と表わされるが，これを $\mathbf{v}_k$ を順次求める形に書換えると

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

となる．(2) の最後の式は (1) の最後から 2 つめのに相当しており，(1) の最後の関係が (2) の最後と矛盾しないためには，(2) を形式的に $m+1$ まで拡張したとき $\mathbf{v}_{m+1}$ が 0 になれば良い．順次求める $\mathbf{v}_k$ の列の $m+1$ 番目が 0 になるためには，$\mathbf{v}_k$ をそれ以前の $\mathbf{v}_{k-1}, \mathbf{v}_{k-2}, \cdots$ と直交させておけば良い．空間次元が m だから $m+1$ 以上の直交するベクトルは出来ないのである．うまいことに，(2) 式は以前作ったベクトルの成分を差し引いてと直交させていると見ることが出来，実際

$$\begin{aligned}
\alpha_i &= \mathbf{v}_i^T H\mathbf{v}_i \\
\beta_i &= \| H\mathbf{v}_i - \beta_{i-1}\mathbf{v}_{i-1} - \alpha_i\mathbf{v}_i \|
\end{aligned} \tag{3}$$

とするとすべての $\mathbf{v}_k$ は互いに直交することがわかっている．ところで，これは前述の反復法の改良に等価になっていてそれゆえ，基底固有状態などを求めるためには，$m-1$ 回の (2) 式の操作をすべて実行しなくても，適宜途中ですでに 3 重対角化された部分だけについてバイセクション法で固有値を調べ，満足できる精度で収束していれば打ち切ることが許される．

固有ベクトルを求めるには，固有値の収束が確認された時点でその 3 重対角化された行列の固有ベクトルを計算しておく．3 重対角行列ともとの行列の違いは表現の基底の取り方にあり，これらの間の変換は行列 V による．したがって，もとの行列表現での固有ベクトルは 3 重対角行列の固有ベクトルの成分 $c_1, c_2, \cdots$ を V の列ベクトル $\mathbf{v}_1, \mathbf{v}_2, \cdots$ に掛けて和を取れば良い．しかしながら，通常数十回かかる収束までの間これらの列ベクトル $\mathbf{v}_1, \mathbf{v}_2, \cdots$ のすべて (数十個) を記憶しておくことは大規模行列では不可能である．最も簡単な解決方法は，$c_1, c_2, \cdots$(これはせいぜい数十個の実数) を記憶しておきもう一度 Lanczos 法の手続きを繰り返して次々に生成される $\mathbf{v}_1, \mathbf{v}_2, \cdots$ に $c_1, c_2, \cdots$ を掛けて足し上げるという操作である (参考文献 4)．lncv1, lncv2 ではこの方法を採用している．

B.2 逆反復法と CG 法

近似的な固有値 E_a がわかっているときにそれに対応する固有ベクトルを求めるのには逆反復法が良く使われる．任意の初期ベクトル $\mathbf{v}_0$ から出発して，$(H - E_a)^{-1}$ を繰り返し掛けるのである．つまり，$\mathbf{v}_0$ を H の固有ベクトルで展開して，

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

とすると，$(H-E_a)^{-1}$ を k 回掛けたとき

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

となり，E_a に最も近い固有値 E_{ja} の成分が他の成分に比べて指数関数的に大きな重みを持つようになる．E_a の精度が良いほど収束は速く，inv1, inv2 に lnc1, lnc2 で求めた固有値を入れると普通 1 回か 2 回で収束する．

$(H-E_a)^{-1}$ を掛けるという操作をそのまま実行しようとすると逆行列を求める必要が生じ，簡単には出来ない．しかしながら

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

を書換えると

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

となり，これは $\mathbf{v}_{k-1}$ から $\mathbf{v}_k$ を求める連立方程式だから解くことは難しくない．

大規模行列による連立方程式を解く方法のひとつに CG 法がある．これは，ひとことでいうと連立方程式 $M\mathbf{x}-\mathbf{b}=0$ を残差 $\mathbf{r}\equiv\mathbf{b}-M\mathbf{x}'$($\mathbf{x}'$は近似解) についての 2 次形式 $f(\mathbf{x}')\equiv(\mathbf{r}, M^{-1}\mathbf{r})$ の極値問題として解こうというものである．ここで (,) は内積を表す．$f(\mathbf{x}')$ を極値に持って行くために，近似的な解 $\mathbf{x}'$ を逐次改良する方法として $f(\mathbf{x}')$ の $\mathbf{x}'$ の関数としての最急勾配方向に $\mathbf{x}'$ を変えてみて，その方向で $f(\mathbf{x}')$ を極値にする地点を新たな近似解とするというのが SD (Steepest Descents) 法である．CG (Conjugate Gradient) 法は SD 法の改良であり，修正方向を最急勾配方向のベクトルからそれ以前の修正方向すべての成分を引き去ったものを選ぶ手法である．

Appendix C. ランダム磁場がある場合の書換え

境界などにさまざまな磁場を加えたときの系の物理量の振る舞いを調べたい場合がある．TITPACK Ver. 2 では，あまりハミルトニアンを一般化すると処理が遅くなったり必要な記憶容量が増えるため，磁場の効果は自明な z 方向の一様磁場を除いては取り入れてない．しかしながら，このような短所があることを承知の上でプログラムを書換えたいというユーザのために，どこをどう直せば良いかを簡単に説明する．なお，Appendix C の内容に限り，筆者自身はテスト・ランを行なっていないので十分な検討の上走らせていただきたい．

ハミルトニアンに加える項は，z 方向のランダム磁場 $H_r = -\sum_k h_k S_k^z$ である．まず h_k のデータを配列 rfield(n) に入れておく．

大規模行列の場合，lnc1/lncv1 による処理においては，ルーチン mltply を書換えるだけで良い．mltply に渡された rfield(n) を使って，10 番の do ループの後にもうひとつ do ループを作り，次のように処理する．

```
do 20 k=1, n
   field=−rfield(k)∗0.5d0
   is=2∗∗(k−1)
   do 20 j=1,idim
      ibit=iand(list1(j), is)
      if(ibit.eq.0)then
         v0(j)=v0(j)−field∗v1(j)
         prdct=prdct−field∗v1(j)∗∗2
      else
         v0(j)=v0(j)+field∗v1(j)
         prdct=prdct+field∗v1(j)∗∗2
      end if
20 continue
```

check1 においても同様に 20 番のループの後で x にランダム磁場のハミルトニアン H_r を掛けて v に足し上げるループを加える．cg1 でも 40 番のループの後で，p にランダム磁場の項を掛けて y に足し上げるループを付ける．その際 40 のループと同様に，逆反復法の常として，掛けられるのはハミルトニアン H_r 自身ではなくてハミルトニアンから近似固有値を差し引いたものであり，40 番のループの eperbd という変数に相当する eperst=E/float(n) を H_r から引いておく必要があることに注意する．

中規模行列では，最初に行列要素を作る elm2 だけの書換えで良い．20 番のループの後で上記と同じように，対角要素の値 field を ibit が 0 かそうでないかに応じて $-$ か $+$ の符号で elemnt(i, ic) (i=1, idim) に加える．

小規模用についても同じく elm3 の 30 番のループの後の新たなループで，対角要素 elemnt(i, i) に field の値を適切な符号で加える．共通に使われるルーチン群 (xcorr, zcorr, orthg など) はそのままで良い．

そのまま走らせられるメイン・プログラムの例をいろいろな状況に応じて作成した．各プログラムで実行している内容は次のとおりである．特に断ってない限り，系は n=16 の 1 次元最近接反強磁性ハイゼンベルクモデル ($S_z^{\rm total} = 0$) である．

Sample 1: lnc1 で固有値を求め lncv1 で基底固有ベクトルを求める． 精度を check1 で調べ，xcorr, zcorr で 2 点相関関数を計算する．

Sample 2: lnc1 で固有値，inv1 で基底固有ベクトルを求める．精度を check1 で調べ，xcorr, zcorr で 2 点相関関数を計算する．

Sample 3: lncv1 で基底，第 1，第 2 励起状態の固有ベクトルを求め，第 2 励起ベクトルの精度チェックをする．

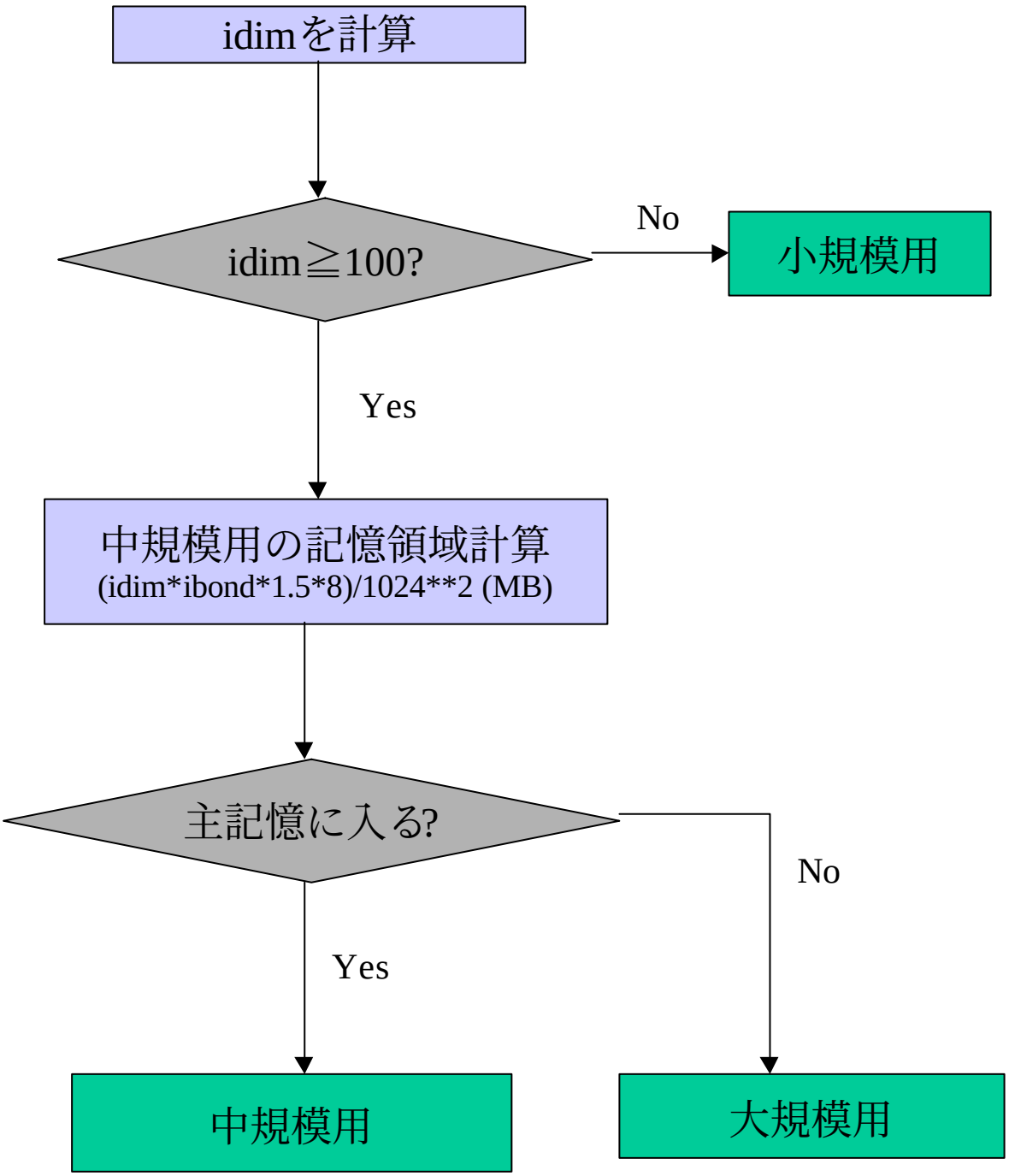
Sample 4: inv1 で第 2 励起状態の固有ベクトルを求め，精度チェックをする．

Sample 5: 固有ベクトルを 2 つの異なる初期条件で求め，縮退度を調べる．

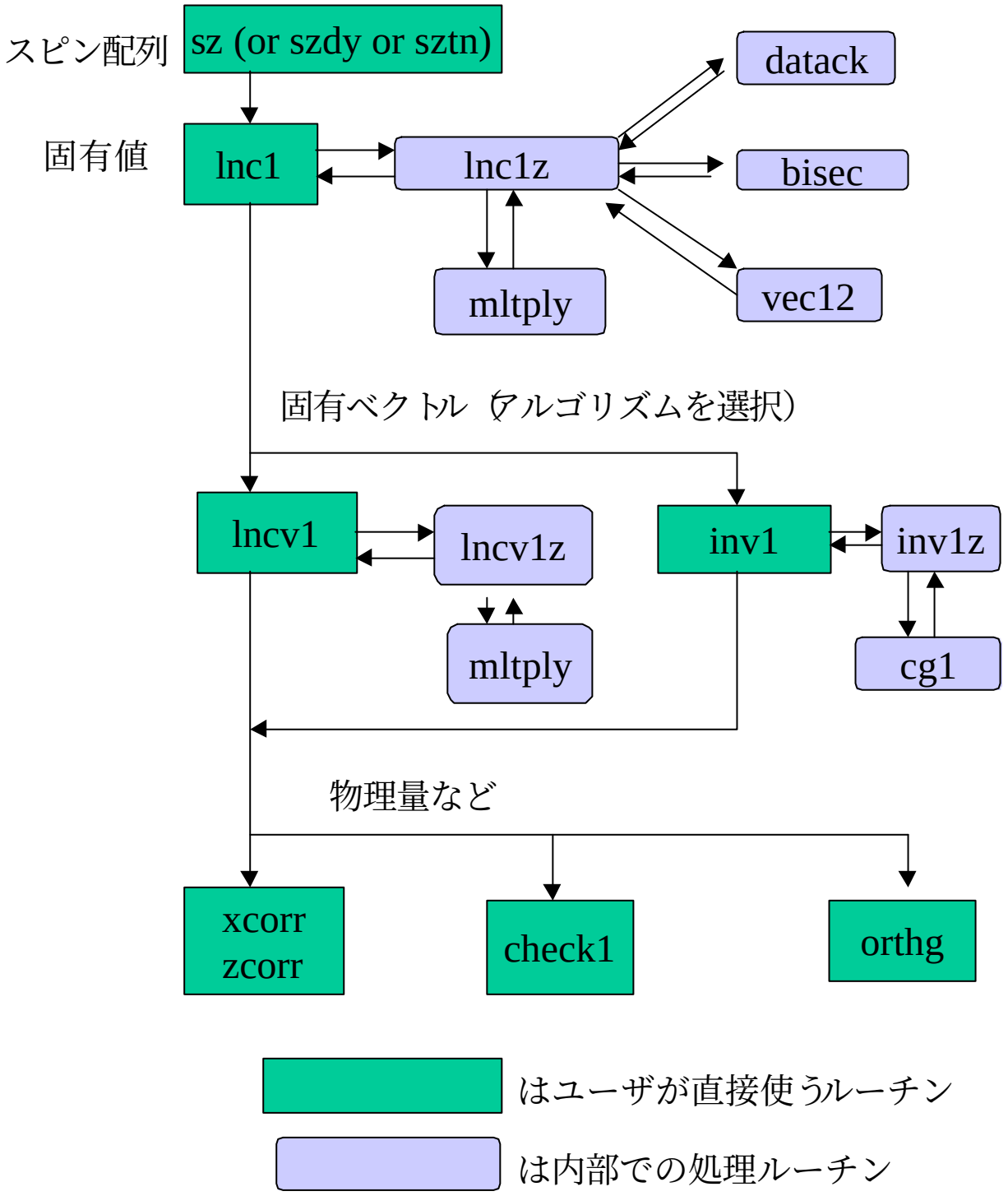
Sample 6 – 10: Sample 1 – 5 と同じ処理を中規模用ルーチンで行なう．ただし，縮退度のチェックは n=15 について実行している．

Sample 11: diag を使って sample 1 と同じ処理を n=8 について行なう．

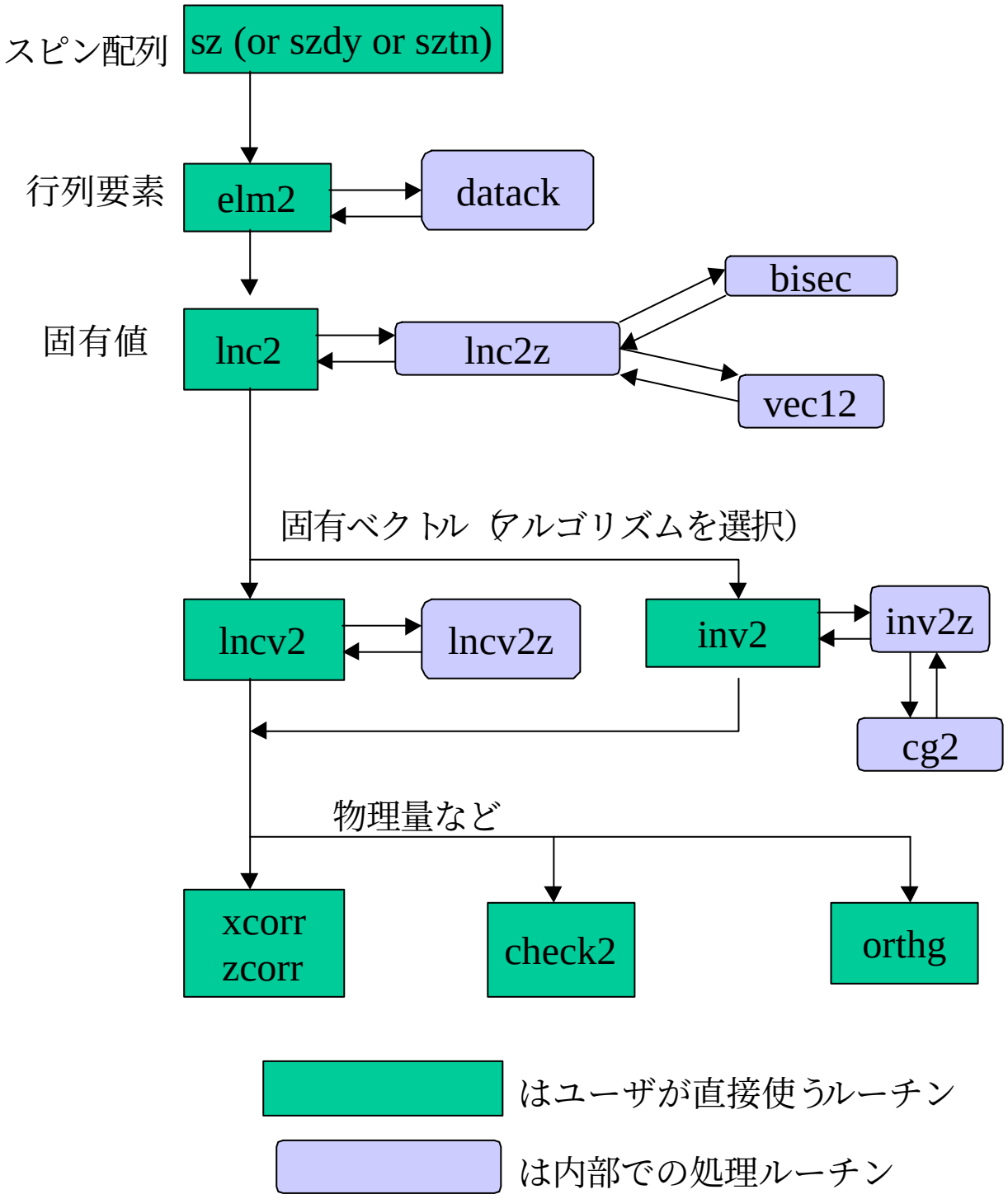
1.2 ルーチン群の選択


idimを計算
idim≧100?
No
小規模用
Yes
中規模用の記憶領域計算
(idim*ibond*1.5*8)/1024**2 (MB)
主記憶に入る?
No
Yes
中規模用
大規模用

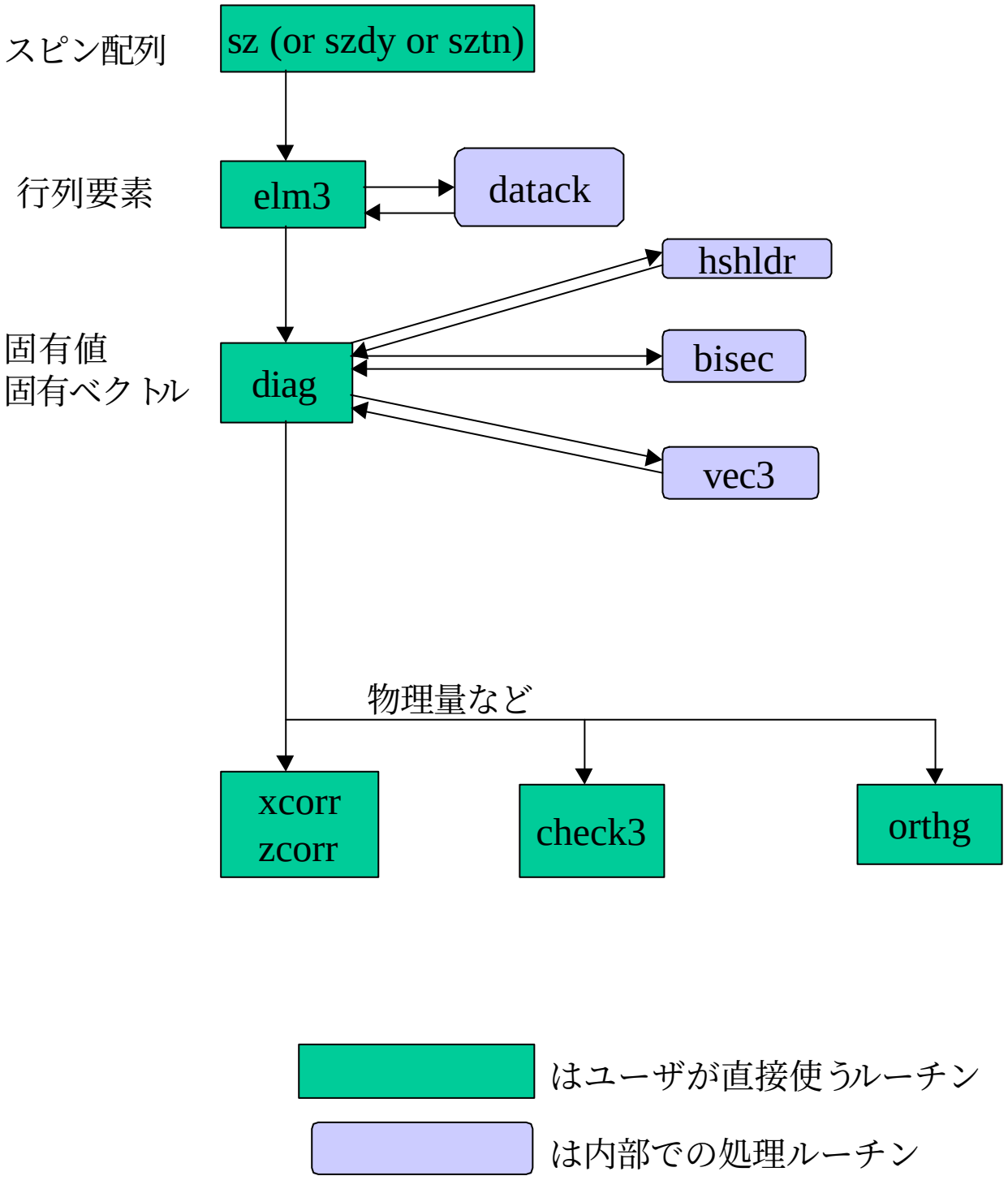
1.3 大規模行列における処理の流れ



1.4 中規模行列における処理の流れ



1.5 小規模行列における処理の流れ



2.1 大規模行列
[2]Lanczos法による固有値の計算
ipair(ibond*2)への図