強相関電子系における超大規模固有値問題 ——地球シミュレータ上でのベクトル並列計算

山田 進[†] 町田昌彦[†] 今村俊幸[††]

強相関電子系の電子状態を求める際に現れる超大規模行列の固有値並列計算手法を提案する．一般に強相関電子系のハミルトニアン行列は直積の形で表現可能であり，本論文ではその構造を利用し，地球シミュレータに代表されるベクトル並列計算機を最大限活用するベクトル・並列計算手法を提案する．この手法を用い，地球シミュレータの 128 ノード（1024 プロセッサ）を利用することで 24 サイトの d-p モデルに対応する約 180 億次元の行列の固有値および固有ベクトルを計算することに成功した．さらに計算性能の評価から，本手法は通信の待ち時間が少なく，またノードごとの計算時間および通信時間がほぼ均等であることを確認した．

Parallel Computation of Large-scale Eigenvalue Problems of Strongly Correlated Electrons on the Earth Simulator

Susumu Yamada,[†] Masahiko Machida[†] and Toshiyuki Imamura[††]

We propose a parallelization technique for solving huge size matrices encountered in calculating electronic structures of strongly correlated electron systems. By utilizing a specific character that these matrices are generally represented by direct product in the Hamiltonian matrix for the strongly correlated electron systems, we develop the technique effective in vector-parallel architecture as the Earth Simulator. Consequently, we obtain both the smallest eigenvalue and the eigenvector of a matrix with about 18 billion-dimension for 24-site problem of "d-p model" on the Earth Simulator. Moreover, it is confirmed that the execution and communication loads of the parallel calculation are almost equally distributed to each node.

1. はじめに

1986 年，ベドノルツとミュラーにより，従来の物性物理の常識を超えた銅酸化物高温超伝導体が発見されて以来[1)]，電子間クーロン反発力が主要な役割を果たす強相関電子系に関する研究が一躍脚光を浴びてきた．しかしながら，いまだ，強相関電子系の物性を確実に予測できる強力な理論的手段は確立されておらず，既知の物理現象さえ，多くの説明が乱立するといった混乱した状況がしばしば見受けられる[2)]（高温超伝導機構が未解決であるということも 1 つの例である）．こうした現状を打開するため，これまでに数多くの数値計算による研究手法が提案されてきたが，それらにより得られた結果も出版される論文ごとに異なるなど，多くの問題をかかえており，シミュレーションサイズを大きくし，かつ高精度で計算することが望まれている．

さて，高温超伝導体の発見により，強相関電子系の研究が著しく進展した一方，その間の計算機の発達も目覚ましく，スーパーコンピュータの計算能力は，現在，様々な科学分野において，これまで理解および取扱い不可能とされてきた問題を解明できるものと期待されている．こうした計算機資源の高速化および巨大化の影響は，もちろん強相関電子系を研究する分野にも大きく波及し，量子モンテカルロ法や厳密対角化法☆[3)] などの手法を発達させ，有限系ながらも強相関電子系の電子状態の一端を垣間見せるほどに成長してきた．

本論文では，この強相関電子系を理解するため発達してきた厳密対角化法に焦点を当て，現時点で最も

† 日本原子力研究所
Japan Atomic Energy Research Institute
†† 電気通信大学
The University of Electro-Communications

☆ 計算機で 1 個または数個の固有値および固有状態（固有ベクトル）を近似的に計算することを厳密対角化と呼ぶことに数学的には違和感があるが，物理分野ではこのように呼ぶのが一般的である．そのため，物理的説明を行っている 1，2 章においては，固有値・固有ベクトルを計算することを（厳密）対角化と呼ぶ．

高速性能であり，かつ巨大メモリを有する地球シミュレータ[4] を利用した場合の超大規模行列の対角化のベクトル並列化手法について述べ，その性能を評価する．

以下，本論文の構成を記す．2 章では，本論文で計算を行った高温超伝導体の基本モデル，すなわち，d-p モデル[3] について記し，強相関電子系の物理的特徴を概説する．なお，本論文の主眼は，地球シミュレータ上での大規模シミュレーション手法であるため，得られた物理的結果を議論するものではないことを注意する．したがって，地球シミュレータを利用し直接的にどれだけ大きいサイズ（サイトの数と解くべきハミルトニアン行列の次元）の固有値問題を開発した並列化手法を用いて解くことができるかという点に注目していただきたい．3 章では，強相関電子系厳密対角化手法で一般に用いられる Lanczos 法について概説する．4 章および 5 章では，地球シミュレータの特長であるベクトルプロセッサのためのベクトル化手法，および共有分散メモリのための並列化手法について説明する．そして 6 章では，実際に地球シミュレータを利用してベクトル・並列計算した際の計算性能を紹介する．

2. d-p モデルとハミルトニアン行列の特徴

酸化物高温超伝導体の発見以後，その電子状態を記述する基本モデルとして，d-p モデルが提案された．以下，この d-p モデルを手短に説明する．モデルハミルトニアンは，

$$\begin{aligned} H_{d-p} = & \sum_{i\sigma} \epsilon_d d^\dagger_{i\sigma} d_{i\sigma} + \sum_{i\sigma} \epsilon_p p^\dagger_{i\sigma} p_{i\sigma} \\ & + \sum_i U_d n_{di\uparrow} n_{di\downarrow} + \sum_i U_p n_{pi\uparrow} n_{pi\downarrow} \\ & + \sum_i V[n_{di} n_{pi} + n_{di-1} n_{pi}] \\ & + t \sum_i [d^\dagger_{i\sigma} p_{i\sigma} + d^\dagger_{i\sigma} p_{i-1\sigma} + H.c.] \quad (1) \end{aligned}$$

と表せる．ここで，ϵ_d，ϵ_p は，銅の d 軌道，酸素の p 軌道の軌道エネルギーであり，U_d および U_p は，d 軌道および p 軌道上でのクーロン反発エネルギーである．この U_d と U_p により，同じサイトに ↑ の電子と ↓ の電子が来る（2 重占有）とエネルギーが，各々，U_d および U_p だけ増加するため，電子（ホール）は 2 重占有をできるだけ避けながらサイト間を渡り歩くことになる．これが，強相関電子系の特徴である．また，V は d-p 軌道間の電荷移動にまつわるクーロンエネルギーであり，t は同じく d-p 軌道間のホッピングエネルギーである．t は，銅サイトと酸素サイトの間の距離が格子系の歪みにより変化するため，動的（電子系が格子系に比べて素早く応答する場合には静的に扱ってもよい）に変化しうる量であり，銅酸化物超伝導体を含めて遷移金属酸化物では，こうした格子変位も電子系に重要な役割を果たしている可能性が強い[2]．ここでは，将来こうした効果を含めるため，t がサイトごとに変動しうる可能性を残し，系の並進対称性によるハミルトニアン行列次数の低減は行わない．こうして得られる d-p モデルのハミルトニアン行列の次元はモデルが一次元，二次元のどちらの場合も**表 1** のような大きさとなる．なお，電子数はアップスピン，ダウンスピンともサイト数の 4 分の 1 としている．

表 1 サイト数と行列の次元
Table 1 Relation between number of sites and dimension of matrix.

サイト数	行列の次元
12	48,400
16	3,312,400
20	240,374,016
24	18,116,083,216
28	1,401,950,721,600

↑ および ↓ の電子数はともにサイト数の 4 分の 1.

表 1 に示したように，サイト数の増加にともない，ハミルトニアン行列の次元は膨大な大きさに増加する．このため，サイト数の増加とともに膨大な次元数を持つ行列の対角化が求められるが，物理的にはエネルギーの低い固有状態が低温で支配的な役割を果たすため，最低あるいは，最低近傍の数個の固有値と固有状態を得るだけで十分であり，その目的に合致した対角化法として Lanczos 法[3),5),6)] が用いられる．

ところで，表 1 が示すように d-p モデルのサイト数を 4 個増加させると行列の次元は約 2 桁増加することから，必要なメモリサイズも同程度増大する．この必要メモリサイズのサイト数依存性は，並列化してもたかだか 2，3 サイトの拡大しか望めないことを意味しており，強相関電子系研究者の並列化に対する意欲を失わせるに十分であった．そのため，これまで大規模問題を並列計算する研究は Fehske ら[7] や Weiße ら[8] により報告されているが，ベクトル化・並列化についての系統的な研究報告は行われていない．

しかしながら，最近現れた 1000 を超えるプロセッサを持ち，テラバイト級のメモリを有する超並列計算機（地球シミュレータなど）は，物理的に意味あるサイト数の拡大を可能にしている．したがって，本論文は並列化手法の提案が主たる目的だが，並列化を行い地球シミュレータ規模の計算機を用いることで，これまでの限界を超えた計算が可能であることを強相関電子系研究者に報告するという性格もあわせ持つことを

付記する．

次に，次章以降で扱う上記モデルについて説明する．本論文で扱う d-p モデルだけでなく，強相関電子系のハミルトニアン行列は，一般に，クーロン反発力項由来の対角項とサイト間トランスファ由来の非対角項とから構成される．ここで，電子のサイト間トランスファに際し，電子のスピン反転が起こらないということに着目すると，非対角要素としては，たとえばアップスピンを持つ電子配置に関係するものだけを用意し，それと単位行列との直積をとることで全電子配置の非対角成分行列を表すことができる．こうして，ハミルトニアン行列は抽象的には，

$$H_{d-p} \to H = (I \otimes A) + (A \otimes I) + D \qquad (2)$$

と記すことが可能となる．このとき，I は A と同じ次元の単位行列，D は対角行列であり，A はアップスピン（ダウンスピン）電子のトランスファ演算子に対して非零要素を与える行列に相当している（そのため A は疎行列である）．こうして，$I \otimes A$ はダウンスピン（アップスピン）電子の配置を固定した場合のアップスピン（ダウンスピン）配置に対する非零要素を与える一方，$A \otimes I$ はその逆の行列を生成する．

このため，Lanczos 法の行列・ベクトル積の計算は上記の直積計算を利用して行えるため，行列 A および対角行列 D の情報を格納しておけば十分であり，H のすべての非ゼロ要素を記憶する必要がない．

次章以降では，上式 (2) のような抽象化した行列の普遍的な並列化手法のみを議論し，d-p モデルの固有値および固有状態に対して得られる物理的結果については別の論文において公表する．

なお，上記の説明は一般の d-p モデルを対象にしており，また式 (1) の酸素サイトを省略し U_d だけをクーロン反発として残すことにより，強相関電子系モデルの代表格であるハバード モデルに帰着できる[3]．そのため，次章以降の計算例では 1 次元 d-p モデルを対象にしているが，今回議論するベクトル化・並列化手法は 2 次元問題やハバード モデルの問題にそのまま適用できることを付け加えておく☆．

3. Lanczos 法

本研究では行列 (2) の最小固有値を求めるのに大規模な対称疎行列の数個の固有値，固有ベクトルを求めるのに向いている Lanczos 法を使用する．

大規模な n 次元実対称疎行列 H に対し**図 1** の方法で $\alpha_0, \alpha_1, \ldots$ および $\beta_1, \beta_2, \ldots$ を求め，m 次元の 3 重対角行列 T_m

初期ベクトル $x \neq \mathbf{0}$ をとる
$\beta_0 = 0$, $v_{-1} = \mathbf{0}$, $v_0 = x_0/||x_0||$
for $k = 0, 1, 2, \ldots, m-1$
　$w = Hv_k - \beta_k v_{k-1}$
　$\alpha_k = (w, v_k)$
　$w = w - \alpha_k v_k$
　$\beta_{k+1} = \sqrt{(w, w)}$, $v_{k+1} = w/\beta_{k+1}$
endfor

図 1 Lanczos 法アルゴリズム
Fig. 1 Algorithm of Lanczos method.

$$T_m = \begin{pmatrix} \alpha_0 & \beta_1 & & & 0 \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-2} & \alpha_{m-2} & \beta_{m-1} \\ 0 & & & \beta_{m-1} & \alpha_{m-1} \end{pmatrix} \qquad (3)$$

が得られ，この行列 T_m の固有値は元の行列 H の（絶対値の大きい）固有値の良い近似になっている[5),6)]．このとき，今回の問題では Hv_k はそのまま計算せず，式 (2) の関係から $Hv_k = Dv_k + (A \otimes I)v_k + (I \otimes A)v_k$ とおき，右辺の各項ごとに計算する．行列 T_m は元の行列 H に比べて小規模な 3 重対角行列なので 2 分法や QR 法でなどの解法で固有値を容易に求めることができ，これにより得られた値が Lanczos 法により求められた行列 H の近似固有値ということになる．この近似固有値は絶対値の大きいものほど高精度で近似することが知られている[5),6)]．

また，この固有値に対応する固有ベクトル z は $V = (v_0, v_1, \ldots, v_{m-1})$ とおき，T_m の固有ベクトルを $y^{(m)}$ とすれば，$z = Vy^{(m)}$ で与えられる．今回扱う問題のサイズは大きく $v_0, v_1, \ldots$ をすべて保存することができない．そのため，実際の計算では，一度 Lanczos 法により 3 重対角行列 T_m を計算し，その固有ベクトル $y^{(m)}$ を求める．その際，T_m の作成に不必要になった v_i は破棄しながら計算していく．その後，Lanczos 法により再び $v_0, v_1, \ldots$ を計算し，先ほど求めた $y^{(m)}$ を利用し固有ベクトル z を計算する．このとき，v_i を求めるごとに $z \leftarrow z + y_i^{(m)} v_i$ を計算し，1 回目の計算同様，必要のなくなった v_i は破棄する．以上の計算では，1 回目と 2 回目の $v_0, v_1, \ldots$ の値をすべて同じにする必要があるため，計算方法はまったく同じにする．また，1 回目で求めた α_k，β_k の値はメモリの使用量は少ないのですべて保存し，2

☆ ただし，並進対称性を用いて行列を低減化した場合には本論文の手法をそのまま適用することはできない．

回目の計算時に利用する．

4. ベクトル化

図 1 にあるように Lanczos 法の計算は，ベクトルの内積（ノルム），ベクトルの線形和，ベクトルの正規化および行列とベクトルの積が大きな計算量を占めている．このうち，内積，線形和，正規化および行列とベクトルの積のうち Dv の計算はそのままでベクトル計算が可能である．一方，$(I \otimes A)v$ および $(A \otimes I)v$ のベクトル計算については行列を完全な形で保持しないため，特別の配慮が必要である．そこで，これらの積のベクトル化手法について説明する．

2 章で述べたように行列 A は疎行列である．通常，疎行列とベクトルの積をベクトル計算する際には行列の格納形式に最内ループを長くできる Jagged Diagonal Storage（JDS）形式を利用するのが良いとされている[9]．JDS を利用した $(I \otimes A)v$ の計算は

```
do l=1,n                    ··· (a)
 nn=(l-1)*n
 do i=1,nnzrow
  do j=jr(i),jr(i+1)-1
   k=j-jr(i)+1
   w(nn+ipt(k))=w(nn+ipt(k))+a(j)*v(nn+jc(j))
  enddo
 enddo
enddo
```

と表せる．ここで `nnzrow`，`jr`，`jc`，`a` および `ipt` はそれぞれ行列 A についての 1 行あたりの非ゼロ要素の個数の最大，JDS 形式の列方向の始まりを表すポインタ，JDS 形式の非ゼロ成分の列位置，JDS 形式の係数，および置換前の行番号である．このとき v および w はそれぞれ `jc` および `ipt` を間接指標ベクトルとしているため，メモリへのアクセスが高速にできない[10]．そのため，等間隔のアクセスになるよう (a) のループが最内となるようにループの順序を交換し，ブロックの大きさである n おきにアクセスするようにする．さらに，ブロックの大きさ n が偶数の場合は，各ブロックの最後に要素を 1 つ追加し，$n+1$ と奇数おきの参照にし，メモリアクセスを高速で行えるようにする[10]．

また，$(A \otimes I)v$ は JDS を用いた場合

```
do l=1,n                    ··· (b)
   do i=1,nnzrow
      do j=jr(i),jr(i+1)-1
        k=j-jr(i)+1
        nn0=(ipt(k)-1)*n
        nn1=(jc(j)-1)*n
        w(nn0+l)=w(nn0+l)+a(j)*v(nn1+l)
      enddo
   enddo
enddo
```

```
do i=1,n
    do j=ir(i),ir(i+1)-1
       av=a(j)
       icc=ic(j)
       do l=1,n
          nn=(l-1)*n0
          w(nn+i)=w(nn+i)+av*v(nn+icc)
       enddo
    enddo
enddo
```

図 2 $(I \otimes A)v$ のベクトル化
Fig. 2 Vectorization of $(I \otimes A)v$.

```
do i=1,n
    nn0=(i-1)*n0
    do j=ir(i),ir(i+1)-1
       av=a(j)
       nn1=(ic(j)-1)*n0
       do l=1,n
          w(nn0+l)=w(nn0+l)+av*v(nn1+l)
       enddo
    enddo
enddo
```

図 3 $(A \otimes I)v$ のベクトル化
Fig. 3 Vectorization of $(A \otimes I)v$.

と表せる．ただし，`nnzrow`，`jr`，`jc`，`a` および `ipt` は前述のものと同じである☆．この計算においては (b) のループを最内ループにすることにより連続ベクトルの参照になるため，メモリからの読み出し，書き込みを高速で行うことができる．

上記の計算方法は，行列ベクトル積の計算における最内ループはブロック数 n になるため，行列 A の格納形式に複雑な JDS を採用する必要はなくなる．そのため，本研究では疎行列の一般的な格納法である Compressed Row Storage（CRS）形式[9] を採用する．その際の $(I \otimes A)v$ および $(A \otimes I)v$ の計算方法をそれぞれ**図 2** および**図 3** に示す．ただし，`n`，`ir`，`ic` および `a` はそれぞれ行列 A の次元，CRS 形式の各行の始まりを表すポインタ，CRS 形式の非ゼロ要素の列位置および CRS 形式における係数とする．また $n0 = n + \mathrm{mod}(n+1, 2)$ とする．このとき，図 2 および図 3 の最内ループがそれぞれ奇数の等間隔および連続のアドレスの参照になっていることが確認できる．

以上のベクトル化の効果を確認するために，実際に地球シミュレータの 1 プロセッサを用い，1 次元 20 サイトの d-p モデル問題のハミルトニアン行列 H

☆ 実際の計算では，ブロックの大きさ n が偶数の場合，バンクの競合を避けるため，各ブロックの最後に要素を 1 つ追加し奇数にして計算する．

表 2 行列ベクトル積の計算時間
Table 2 Execution time of matrix-vector multiplication.

a) $(I \otimes A)v$

方法	計算時間（秒）	FLOPS ($\times 10^6$)	V. OP RATIO	BANK CONF
JDS	34.124	1112.2	99.39	1.2489
CRS	8.733	4346.1	99.42	0.0000

b) $(A \otimes I)v$

方法	経過時間（秒）	FLOPS ($\times 10^6$)	V. OP RATIO	BANK CONF
JDS	36.442	1041.5	99.46	1.0530
CRS	8.688	4368.5	99.42	0.0000

V. OP RATIO··· ベクトル演算率（%）
BANK CONF··· バンクの競合時間（秒）

（240,374,016 次元）とベクトルの積を上記の 2 つの方法で 10 回計算した際の計算時間，1 秒間に計算された浮動小数点データ演算の回数（FLOPS 値）およびベクトル演算率を**表 2** に示す．この表において，比較する 2 つの方法を JDS，CRS と記述しているが，この比較の本質は行列の格納形式ではなく，最内ループの計算方法であることに注意が必要である．

この結果から，どちらの計算方法でもベクトル演算率は 99%を超えているが，JDS を用いた方法では計算時間が多く FLOPS 値が低いことが確認できる．また，バンクの競合が生じていることも確認できる．一方，ループ順序を交換し CRS を採用した方法では，JDS の方法のおよそ 4 分の 1 以下の計算時間であり，そのため FLOPS 値は 4 倍以上になる．また，バンクの競合の発生は確認できない．このことから，ループ順序を交換し，連続または奇数等間隔のメモリアクセスにすることはベクトル計算に有効であると結論付けられる．

5. 並 列 化

地球シミュレータは 8 個のプロセッサで 1 つのノードを構成している．そのため，効率の良い並列計算を行うためには，ノード内では通信を行わない共有メモリ用の並列化を行う必要がある．このノード内の並列化には地球シミュレータ用の並列化指示オプションを利用した自動並列化機能を利用する．また，複数のノードを利用する並列化にはノード間で通信を行う分散メモリ用の並列化を行う必要がある．この並列化のためのノード間通信には MPI（Message Passing Interface）を利用する．

この章では，分散メモリ用並列化のためのデータの分割および通信の方法および共有メモリ用並列化の方法について説明する．

5.1 データの分割

ハミルトニアン行列 H の表現方法 (2) から対角成分 D およびベクトル v は要素数が n の n 個ブロックで構成されていることが分かる．大規模な問題，つまり n が大きい場合，すべての要素をすべてのノードで重複して格納することはできない．そのため，要素を分割し各ノードに分散して格納することが必要になる．今回の計算では行列とベクトルの積の計算ではブロック単位で計算を行うため，分割の単位は要素ではなくブロックとする．このとき，分割方法としてはブロック数 n がノード数 n_p で割り切れる場合は $n_d = n/n_p$ 個ずつ分割する．また，割り切れない場合は

(1) ノード番号 0 から $\mathrm{mod}(n-1, n_p)+1$ まで $\frac{n+(n_p-\mathrm{mod}(n,n_p))}{n_p}$ 個を格納，残りに $\frac{n+(n_p-\mathrm{mod}(n,n_p))}{n_p}-1$ 個を格納する．
(2) ノード番号の小さい順に $n_d = \frac{n+(n_p-\mathrm{mod}(n,n_p))}{n_p}$ 個ずつ格納していく．

の 2 通りの分割方法が考えられる．格納方法 (1) ではノード間のブロック数の不均衡をできるだけ少なくしているが，格納方法 (2) では第 n_p-1 ノードのブロック数が他のノードのものと比べ最大で n_p-1 個少なくなる可能性がある．しかし，どちらの格納方法でも，$\frac{n+(n_p-\mathrm{mod}(n,n_p))}{n_p}$ 個のブロックを計算するノードがあり，1 つのブロックあたりの計算量が同じと仮定すると，どちらの格納法を用いても並列計算時間は等しくなることが予想される．また，格納方法 (2) では，分割前に j 番目のブロックが，分割後には第 $(j-1)/n_d$ ノードの $\mathrm{mod}(j-1, n_d)+1$ 番目のブロックであると容易に表すことができる．以上のことから今回は格納方法 (2) を採用する．

また，行列 A は対角行列 D と比較し要素数は少ないため，すべてのノードで行列 A のすべての情報を持つことにする．

5.2 ノード間並列

図 1 にあるように Lanczos 法の計算は，行列とベクトルの積，ベクトルの内積（ノルム）計算，線形和および正規化が大きな計算量を占めている．このうち，ベクトルの線形和および正規化は，各成分が独立に計算できるため通信などの並列処理のための追加の処理は不必要である．ベクトルの内積は，それぞれのノードに対応する内積値を計算してから，ノード間通信を実行し，全内積値を求める．また，ベクトルと行列の積の計算のうち Dv および $(I \otimes A)v$ の計算は D および v をブロックで分割しており，またすべてのノー

表 3 一般的な方法の通信量
Table 3 Communication amount of conventional method.

ノード数	通信回数	通信量（要素数）	
		合計（$\times 10^9$）	平均（$\times 10^6$）
32	226	48.837	216.093
48	368	54.346	147.679
64	524	57.660	110.038
96	834	62.695	75.174
128	1,170	66.122	56.514

表 4 一般的な方法の計算量
Table 4 Calculation amount of conventional method.

ノード数	計算量（積の回数）（$\times 10^9$）		
	最大	最小	平均
32	5.949	4.540	5.317
48	4.002	2.893	3.544
64	3.036	2.070	2.658
96	2.091	1.299	1.772
128	1.563	0.937	1.329

ドで A の情報を持っているため，通信を行わず並列計算できる．

しかし，$(A \otimes I)v$ の計算に関しては，各ノード自身で格納しているベクトル v の値のみでは計算ができない．そのため，計算の前にノード間通信によって他のノードにある必要な v の要素を受信する必要がある．そこで，本研究では，単位行列との直積という行列の形を考慮した並列計算方法を提案する．また一般的な行列・ベクトル積の計算方法との比較を行う．

5.2.1 一般的な方法

通常，行列とベクトルの積では通信を行いベクトル v を再構成する．また，行列が疎行列である場合は，計算に必要になるデータのみを通信し，通信時間を削減する．このとき，計算に必要となるデータはノード自身が担当している分割行列において，少なくとも 1 つが非ゼロ要素である列に対応しているベクトルの要素である．

本研究で対象にしている 24 サイトの場合の通信量を**表 3** に示す．ノード数が増えるに従い，合計の通信量および通信回数がともに増加していることが確認できる．また $(A \otimes I)v$ の計算量（行列の成分とベクトル成分の積の回数）について**表 4** に示す．この表からノードごとの計算量のばらつきが大きいことが確認でき，効率的な並列計算が困難であると予想される．

また，ベクトルを再構成する際に追加される要素数を**表 5** に示す．この表から分かるように，今回の問題では，受信するデータ量が大きく，ベクトルの再構成を行うと大量のメモリを消費するため，再構成は行わず，データを受信するごとにそのデータを利用し，計算することにする．そこで**図 4** のように，要素がすべて 0 である列を除いた部分行列を CRS 形式で格納する方法を採用する．これにより，ノード番号 `ip` -1 からデータを受信した際の計算を**図 5** のように行うことができる．このとき，`nu`，`nb`，`nd` および `rbuf`$(*, \mathtt{iq})$ はそれぞれ，ノードが担当している要素数，ブロック数，5.1 節の n_d，およびノード番号 `ip` -1 から受信したベクトル成分である．

表 5 追加要素数
Table 5 Number of additional elements.

ノード数	追加要素数（$\times 10^9$）			ノード自身の要素数（$\times 10^9$）
	最大	最小	平均	
32	2.305	0.462	1.526	0.566
48	1.661	0.322	1.132	0.378
64	1.282	0.236	0.901	0.283
96	0.971	0.168	0.653	0.189
128	0.767	0.130	0.517	0.142

5.2.2 提案する方法

ベクトル v は要素数が n の n 個のブロックで構成されているため，

$$v = (v_{1,1}, v_{1,2}, \ldots, v_{1,n}, v_{2,1}, \ldots, v_{n,n})^T$$

と表せる．また同様に w を

$$w = (w_{1,1}, w_{1,2}, \ldots, w_{1,n}, w_{2,1}, \ldots, w_{n,n})^T$$

と表すと，行列とベクトルの積 $w = (A \otimes I)v$ は

$$W = VA^T$$

と表せる．ただし

$$V = (v_1^T, v_2^T, \ldots, v_n^T),\ W = (w_1^T, w_2^T, \ldots, w_n^T)$$

であり，v_i および w_i はそれぞれ

$$v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n}),\ w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,n})$$

である．このとき，V を行方向に分割することにより並列計算が可能である．そこで，次の計算方法を提案する．ここで np および nb をそれぞれ並列計算時のノード数およびそのノードが担当しているブロック数とする．

(1) 前節で説明したように rank k ($k = 0, 1, 2, \ldots, np-1$) のノードは v の値として

$$\begin{aligned} V_k &= (v_{n_k+1}, v_{n_k+2}, \ldots, v_{n_k+nb})^T \\ &= (v_{n_k+1,1}, v_{n_k+1,2}, \ldots, v_{n_k+1,n}, \\ &\qquad v_{n_k+2,1}, \ldots, v_{n_k+nb,n})^T \end{aligned}$$

を格納している．このとき $n_k = k \times n_d$ である（n_d は 5.1 節を参照）．この V_k を np 個の小ブロックに分割し，同じ位置の小ブロックを集めるように通信を行い，

$$\begin{aligned} vtmp_k = (&v_{1,n_k+1}, v_{1,n_k+2}, \ldots, v_{1,n_k+nb}, \\ &v_{2,n_k+1}, \ldots, v_{n,n_k+nb})^T \end{aligned}$$

を作成する．

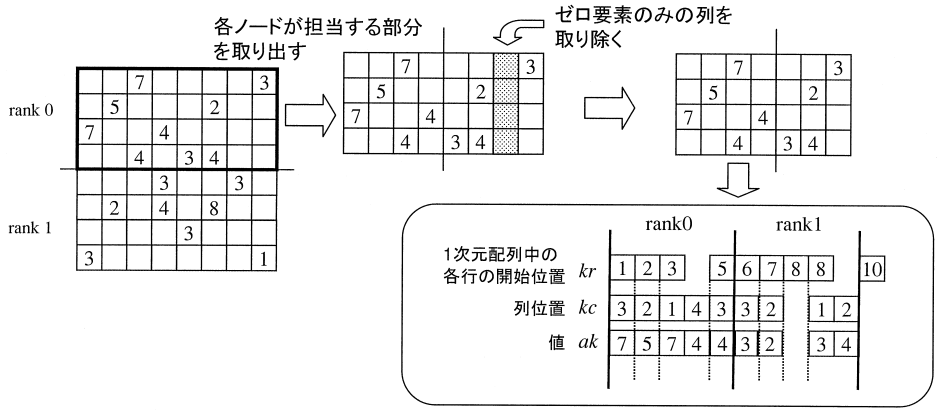

図 4 Compressed Row Storage 形式による行列の格納法（例：行列サイズ 8×8，ノード数 2 の場合）
Fig. 4 Compressed Row Storage Format for the matrix (Example: Matrix size: 8×8, Nsumber of nodes: 2).

```
do ip=1,np
  ii=(ip-1)*nb
  do j=1,nb
    jj=(j-1)*nd
    do l=kr(ii+j),kr(ii+j+1)-1
      kk=(kc(l)-1)*nu
      av=ak(l)
      do k=1,nu
        vv1(jj+k)=vv1(jj+k)+rbuf(kk+k,ip)*av
      enddo
    enddo
  enddo
enddo
```

図 5 $(A \otimes I)v$ のアルゴリズム（一般的な方法）
Fig. 5 Algorithm of $(A \otimes I)v$ (Conventional method).

(2) **図 6** のように各ノードで並列に乗算を行い，

$$wtmp_k = (w_{1,n_k+1}, w_{1,n_k+2}, \ldots, w_{1,n_k+nb}, w_{2,n_k+1}, \ldots, w_{n,n_k+nb})^T$$

を計算する．

(3) 得られた $wtmp_k$ の値を rank k のノードの w の値が

$$W_k = (w_{n_k+1,1}, w_{n_k+1,2}, \ldots, w_{n_k+1,n}, w_{n_k+2,1}, \ldots, w_{n_k+nb,n})^T = (w_{n_k+1}, w_{n_k+2}, \ldots, w_{n_k+nb})^T$$

となるように通信する．

この計算方法おいて行列とベクトルの積の前後に，nb^2 個の成分を自身以外の $np-1$ 個のノードに通信する必要がある．今回扱う 24 サイト問題の場合の通信量を**表 6** に示す．この表からノード数が増加しても通信の合計はほとんど増加せず，一般的な方法より少ないことが確認できる．しかし，通信回数は増加し，

```
do i=1,n
  ii=(i-1)*nb
  do j=ir(i),ir(i+1)-1
   jj=(ic(j)-1)*nb
   av=aa(j)
   do kk=1,nb
     wtmp(ii+kk)=wtmp(ii+kk)+vtmp(jj+kk)*av
   enddo
  enddo
enddo
```

図 6 $(A \otimes I)v$ のアルゴリズム（提案した方法）
Fig. 6 Algorithm of $(A \otimes I)v$ (proposed method).

表 6 提案した方法の通信量
Table 6 Communication amount of proposed method.

ノード数	通信回数	通信量（要素数）	
		合計（$\times 10^9$）	平均（$\times 10^6$）
32	1,984	35.100	17.691
48	4,512	35.477	7.863
64	8,064	35.666	4.423
96	18,240	35.855	1.966
128	32,512	35.949	1.106

一般的な方法と比較し，128 ノードで約 28 倍多い．ところが，1 回あたりの平均の通信量が倍精度のデータで 10^6 個以上と多いため，通信時間に関しては通信開始のためのオーバヘッドより実際に通信を行っている時間の方が支配的であると思われる．また，ノードごとの計算量を**表 7** に示す．この表からすべてのノードの計算量はほぼ同じであり，計算量の不均衡のない効率的な並列計算が行えることが予想される，また，一般的な方法よりも通信量の少ない提案した方法の方が優れた並列計算性能になることが予想される．

5.3 通信方法

一般的な方法では通信すべきデータのアドレスは不

表 7 提案した方法の計算量
Table 7 Calculation amount of proposed method.

ノード数	計算量（積の回数）（$\times 10^9$）		
	最大	最小	平均
32	5.318	5.282	5.317
48	3.546	3.490	3.544
64	2.660	2.584	2.658
96	1.773	1.657	1.772
128	1.330	1.254	1.329

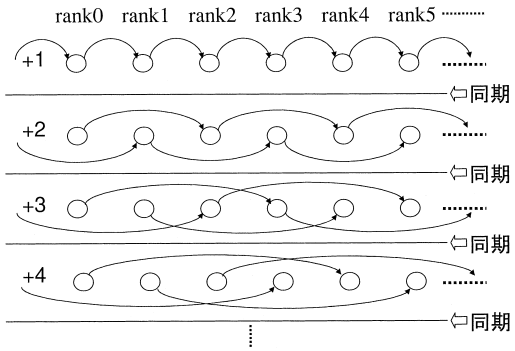

図 7 データの通信方法
Fig. 7 Method of communication.

連続のため，アドレスが連続になるように適当な配列に格納し，通信する方法を採用する．一方，提案した方法についても一般的な方法と同様に，通信するデータをアドレスが連続になるように適当な配列に格納し，通信する方法を採用する．

また，どちらの方法も通信を行う際には，1 対 1 通信（MPI_ISEND および MPI_IRECV）を用いて，**図 7** のように各ノードから同期してすべてのノードから等距離のランクのノードにデータを送るようにする．この通信方法では，他のノードと送受信先が競合することがなく，問題が大規模になっても効率良くデータの通信が行えることが期待できる．

6. 地球シミュレータでの性能評価

上で述べた強相関電子系に現れる大規模な行列に対する Lanczos 法のベクトル化・並列化の性能を評価するため，地球シミュレータを用いて 24 サイト問題に現れる行列の最小固有値およびその固有ベクトルを並列計算する．この行列は，行列 A の次元および要素数はそれぞれ 134,596 および 1,264,032 であり，行列 H の次元は 18,116,083,216 である．この問題を 96 ノード（768 プロセッサ）および 128 ノード（1024 プロセッサ）の 2 通りで計算する．また，Lanczos 法により作成される 3 重対角行列の次元は 200 とする．この 3 重対角行列に対する固有値および固有ベクトルの計算は

表 8 経過時間および誤差
Table 8 Elapsed time and error.

a) 一般的な方法
a) Conventional method

ノード数	経過時間（sec）			相対誤差
	固有値	ベクトル	合計	（$\times 10^{-6}$）
96	717.97	716.20	1434.17	1.018
128	610.54	607.13	1217.67	2.141

b) 提案した方法
b) Proposed method

ノード数	経過時間（sec）			相対誤差
	固有値	ベクトル	合計	（$\times 10^{-6}$）
96	155.61	153.92	309.53	1.018
128	105.49	102.71	208.20	2.141

1 回だけであり，Lanczos 法の計算時間と比較し少ないため，今回は地球シミュレータ用のプログラムは作成せず，日本原子力研究所計算科学技術推進センターで公開している並列数値計算ライブラリ PARCEL [12] の 2 分法および逆反復法のルーチンを利用する．

ところで，地球シミュレータはローカルメモリ（LMEM）およびグローバルメモリ（GMEM）と呼ばれている 2 種類のメモリ空間を実装しており，通信に MPI を用いる場合には LMEM より GMEM に格納されているデータの方が高速に通信できる[13]．そのため，今回の計算では通信するデータは GMEM に割り付ける．

一般的な方法および提案した方法により，最小固有値および固有ベクトルを求めた際の経過時間および相対誤差を**表 8** に示す．ここでの誤差の値は $||\lambda x - Hx||_2/||x||_2$ を採用する．ただし，λ および x はそれぞれ求めた固有値および固有ベクトルである．また，行列成分の作成などを含めたすべての計算におけるベクトル演算率，FLOPS 値およびメモリ使用量（GMEM を利用した場合には正確なメモリ使用量が計測できないため LMEM を利用して同様の計算したときのメモリの使用量）を**表 9** に示す．さらに，地球シミュレータの簡易性能解析機能[10] を利用し得られたノード単位の固有値，固有ベクトル計算を行う際の通信時間（通信手続きの実行に要した時間，同期のための時間などを含んだすべての通信時間），および通信の待ち時間（通信を行うまでの待ち時間および同期待ちに要した経過時間の合計）の最大，最小，平均をそれぞれ**表 10** および**表 11** に示す．

一般的な方法は表 5 に示したように受信するデータ量が多く，すべてのデータを同時に持つことができないため，受信するごとにそのデータを用いて計算を行う．その後，そのデータを破棄し次の通信を行うよ

表 9 プログラム実行解析情報
Table 9 Information about program execution.
a) 一般的な方法
a) Conventional method

ノード数	V. OP RATIO	FLOPS ($\times 10^9$)	メモリ使用量 (GB)
96	96.807	233.086	924.578
128	96.288	276.946	951.234

b) 提案した方法
b) Proposed method

ノード数	V. OP RATIO	FLOPS ($\times 10^9$)	メモリ使用量 (GB)
96	98.647	1058.770	1109.281
128	98.518	1561.437	1117.547

V. OP RATIO· · · ベクトル演算率（%）

表 10 通信時間
Table 10 Communication time.
a) 一般的な方法
a) Conventional method

ノード数	通信時間（sec）		
	最大	最小	平均
96	1298.023	1200.533	1242.756
128	1118.167	1042.866	1071.916

b) 提案した方法
b) Proposed method

ノード数	通信時間（sec）		
	最大	最小	平均
96	188.795	180.433	181.175
128	114.952	108.268	108.832

表 11 通信の待ち時間
Table 11 Wating time for communication.
a) 一般的な方法
a) Conventional method

ノード数	待ち時間（sec）		
	最大	最小	平均
96	1249.832	952.889	1072.837
128	1078.173	851.624	935.733

b) 提案した方法
b) Proposed method

ノード数	待ち時間（sec）		
	最大	最小	平均
96	32.891	19.212	23.143
128	24.623	14.690	15.837

うにしている．そのため，すべてのデータを受信して計算する提案した方法よりメモリ使用量は少なくなる（表 9 参照）．しかし，一般の方法では 1 組でも通信が起こるとすべてのノードが通信を待たなくてはならないため，表 11 に示されているように，大量の通信の待ち時間が生じてしまう．一方，提案した方法は上記の理由でメモリ使用量は多くなるが，通信の同期はすべてのデータを受信した後にのみ行えばよいため，表 11 のように通信の待ち時間は少ない．

表 12 演算時間
Table 12 Floting-point operation time.
a) 一般的な方法
a) Conventional method

ノード数	演算時間（sec）		
	最大	最小	平均
96	233.247	135.829	191.053
128	174.250	99.004	145.214

b) 提案した方法
b) Proposed method

ノード数	演算時間（sec）		
	最大	最小	平均
96	128.302	119.903	127.559
128	98.829	92.152	98.272

また，ノード単位の経過時間と通信時間の差を演算を行った時間として最大，最小，平均を**表 12** に示す．この結果から，提案した方法の方が演算時間が少なく，また演算時間が均一であることが確認できる．

以上のことから，通信の待ち時間が少ないことおよび計算が均等に分割されていることが提案した方法が一般的な方法より早く計算できることの理由であると結論付けられる．

7. ま と め

本論文では，地球シミュレータにおいて強相関電子系の計算に現れる大規模な行列の固有値計算を効率的に行える並列計算方法を Lanczos 法を基に提案した．この方法は並列計算時の通信量が少なく，また各ノードの計算量がほぼ均一になる通信および計算方法を採用している．本方法が地球シミュレータを利用した実際の数値計算から有効であることを確認した．

今後は精度についての調査を行う．さらに，開発した解法を用いて実際の強相関電子系の問題を計算し，物理現象を調査する予定である．

謝辞 本研究を行うにあたり，24 サイト d-p モデル厳密対角化の物理的意義をご教授していただいたペンシルベニア大学（現米国オークリッジ研究所）の江上教授，P.Piekarz 博士および厳密対角化手法についての議論をしていただいた堀田主任研究員（原研先端研）に感謝いたします．また，地球シミュレータ利用に関してご協力いただいた地球シミュレータセンターの方々および叶野氏（原研 CCSE）に深く感謝いたします．さらに，貴重なコメントをいただきました査読者の方々に感謝いたします．

参 考 文 献

1) Bednorz, J.G. and Müller, K.A.: Possible High Tc Superconductivity in the Ba-La-Cu-O System, *Z. Phys.*, B64, 189 (1986).
2) Tachiki, M., Machida, M. and Egami, T.: Vibronic Mechanism of High-Tc Superconductivity, *Phys. Rev.*, B67, 174506 (2003).
3) Dagotto, E.: Correlated Electrons in High-Temperature Superconductors, *Rev. Mod. Phys.*, Vol.66, No.763 (1994).
4) 地球シミュレータセンターホームページ．http://www.es.jamstec.go.jp
5) Cullum, J.K., et al.: *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.1: Theory*, SIAM (2002).
6) 森　正武，杉原正顕，室田一雄：岩波講座 応用数学［方法 2］線形計算，岩波書店 (1994).
7) Fehske, H., et al.: Exact diagonalization results for strongly correlated electron-phonon system, *Proc. NIC Symposium 2001*, Rollnik, H., et al. (Ed.), NIC Series, Vol.9, pp.259–269, John von Neumann Institute for Computing, Jülich (2002).
8) Weiße, A., et al.: Exact diagonalization study of spin, orbital, and lattice correlations in CMR manganites, *High Performance Computing in Science and Engineering 2002*, Krause, E., et al. (Eds.), pp.157–167, Springer, Heidelberg (2002).
9) Barrett, R., et al.: *Templates for the solution of linear systems: Building block for iterative methods*, SIAM (1994).
10) 日本電気株式会社：FORTRAN90/ES プログラミングの手引，日本電気株式会社 (2002).
11) 日本電気株式会社：FORTRAN90/ES 並列処理機能利用の手引，日本電気株式会社 (2002).
12) 並列数値計算ライブラリ PARCEL ホームページ．http://parcel.koma.jaeri.go.jp
13) 地球シミュレータセンターホームページ Message Passing Interface．http://www.es.jamstec.go.jp/esc/jp/Programming/message.html

(平成 15 年 10 月 8 日受付)
(平成 16 年 2 月 10 日採録)

山田　進（正会員）

1970 年生．1998 年東北大学大学院情報科学研究科博士課程修了．同年東北大学大学院工学研究科寄附講座教員．1999 年より日本原子力研究所博士研究員．2000 年より日本原子力研究所研究員．現在に至る．数値計算および並列計算に関する研究に従事．博士（情報科学）．日本応用数理学会，日本計算工学会，日本原子力学会各会員．

町田　昌彦

1963 年生．1991 年東北大学理学研究科博士課程中退．同年（株）富士通入社．同年 10 月より，日本原子力研究所・富士通外来研究員．1996 年富士通退社後，日本原子力研究所研究員．2000 年から 2001 年にかけて米国アルゴンヌ国立研究所研究員．2002 年より日本原子力研究所副主任研究員に昇任．現在に至る．なお，2002 年より産業総合研究所客員研究員を兼ねる．超伝導物性を中心に計算物理学研究に従事．博士（工学）．日本物理学会，日本原子力学会，アメリカ物理学会各会員．

今村　俊幸（正会員）

1969 年生．1996 年京都大学大学院工学研究科応用システム科学専攻博士後期課程単位認定退学．同年日本原子力研究所入所．計算科学技術推進センターにて途切れのない思考を支援する並列処理基本システム STA の開発に従事．2001 年から 2002 年までシュツットガルト大学 HLRS にて招聘研究員．2003 年より電気通信大学講師．現在に至る．HPC とその周辺ソフトウェア，数値計算における並列・分散処理の研究に従事．博士（工学）．平成 11 年日本応用数理学会論文賞，同年石川賞企業部門受賞．日本応用数理学会，SIAM 各会員．