数 値 解 析 37－14
（１９９１．７．１７）

直接法による連立１次方程式の並列解法アルゴリズム

望月義幸　平田博章　西村明夫　西澤貞次

松下電器産業株式会社　情報通信関西研究所

直接法による連立１次方程式の解法として，多枢軸列同時消去計算に基づくガウス消去法とガウス・ヨルダン法の並列化手法の提案とその評価を行う．本手法は，分散共有メモリ型並列計算機を対象としたもので，理論値による評価から，クラスタ化した方が，処理時間，加速率の点で有利であることが解った．また，問題規模がプロセッサ数に較べ大きいときは，枢軸数を増やすことが処理の高速化につながり，この仮定が成り立たないときは，問題規模とプロセッサ数に応じて枢軸数を可変的に使うことが必要であることが解った．

Parallel Algorithms for the Solution of Linear Systems of Equations by Direct Methods

Yoshiyuki MOCHIZUKI　Hiroaki HIRATA　Akio NISHIMURA　Teiji NISHIZAWA

Matsushita Electric Industrial Co.,Ltd.
Kansai Information and Communications Research Laboratory
1006,Oaza Kadoma,Kadoma-Shi, Osaka 571,Japan

This paper proposes and evaluates parallelized, multi-pivot simultaneous elimination calculation based Gauss elimination and Gauss-Jordan elimination methods for the solution of linear systems of equations by direct methods. These methods were developed for a distributed shared memory parallel computer. From a theoretical evaluation, it was decided that clusterization was more advantageous for reasons of processing time and acceleration rate. Furthermore, it was determined that, when the scale of the problem is large in relation to the number of processors, the processing speed increases with an increase in the number of pivots. When this is not the case, it was determined that it is necessary to vary the number of pivots with respect to the scale of the problem and the number of processors.

1　はじめに

連立１次方程式の解法として，直接法は確固たる地位を築いている．その一方で，並列処理技術の重要性は，近年ますます増大している．そのような点から，連立１次方程式の直接法に基づく並列解法について考えることは，重要な課題であると言える．従って，本報告書では，分散共有メモリ型並列計算機を対象とした多枢軸列同時消去計算に基づくガウス消去法とガウス・ヨルダン法の並列化手法の提案を行うことをその目的とする．

多枢軸列同時消去計算は，同時消去を行う枢軸数を２枢軸以上に拡張したもので，それをガウス消去法，及びガウス・ヨルダン法に適用した．２枢軸列同時消去計算に基づくガウス消去法[2),3)]や，LU分解の立場からの多枢軸列同時消去計算に基づくガウス消去法と同様のアルゴリズム[1)]については既に報告されているが，そこでは，並列化に関する考察は行われていない．

2　多枢軸列同時消去計算

以下の各節において，与えられた連立１次方程式は，$Ax=b$，$A=(a_{ij})$，$1\leq i,j\leq n$，$x=(x_1,x_2,\cdots,x_n)^t$，$b=(b_1,b_2,\cdots,b_n)^t$とし，第ｒ列までの消去が行われたときの係数行列を$A^{(r)}=(a_{ij}^{(r)})$，既知数ベクトルを$b^{(r)}=(b_1^{(r)},b_2^{(r)},\ldots,b_n^{(r)})^t$で表すものとする．また，同時消去を行う枢軸数は，ｋ（≧１の整数）とする．

2．1　ガウス消去法

後退代入の部分については，通常のガウス消去法と同じなので，ここでは，多枢軸同時消去計算に基づくガウス消去法の前進消去部分のアルゴリズムについて説明する．

［処理A_1］．係数行列の第pk+1行の各要素と既知数ベクトルｂの第pk+1成分を$a_{pk+1,pk+1}^{(pk)}$で割る．つまり，pk+2≦ｊ≦ｎとして，

$$a_{pk+1,j}^{(pk+1)}=a_{pk+1,j}^{(pk)}/a_{pk+1,pk+1}^{(pk)},$$
$$b_{pk+1}^{(pk+1)}=b_{pk+1}^{(pk)}/a_{pk+1,pk+1}^{(pk)}. \qquad (2.1)$$

但し，枢軸の選択は行い，$a_{pk+1,pk+1}^{(pk)}\neq 0$．

t=2,3,･･･,kとして，［処理A_t］を定義する．

［処理A_t］．係数行列の第pk+t行の各要素及び，既知数ベクトルｂの第pk+t成分に対して次の計算を行う．但し，pk+t≦ｊ≦ｎとする．

$$Reg_{pk+t}^{(0)}=a_{pk+t,pk+1}^{(pk)},$$
$$Reg_{pk+t}^{(1)}=a_{pk+t,pk+2}^{(pk)}-Reg_{pk+t}^{(0)}a_{pk+1,pk+2}^{(pk+1)},$$
$$\cdots,$$
$$Reg_{pk+t}^{(t-2)}=a_{pk+t,pk+t-1}^{(pk)}-\sum_{m=1}^{t-2}Reg_{pk+t}^{(m-1)}a_{pk+m,pk+t-1}^{(pk+m)} \qquad (2.2)$$

$$a_{pk+t,j}^{(pk+t-1)}=a_{pk+t,j}^{(pk)}-\sum_{m=1}^{t-1}Reg_{pk+t}^{(m-1)}a_{pk+m,j}^{(pk+m)}. \qquad (2.3)$$

$$b_{pk+t}^{(pk+t-1)}=b_{pk+t}^{(pk)}-\sum_{m=1}^{t-1}Reg_{pk+t}^{(m-1)}b_{pk+m}^{(pk+m)}. \qquad (2.4)$$

次に，係数行列の第pk+t行の各要素，既知数ベクトルｂの第pk+t成分を第pk+t行の対角要素で割る．つまり，pk+t+1≦ｊ≦ｎとして，

$$a_{pk+t,j}^{(pk+t)}=a_{pk+t,j}^{(pk+t-1)}/a_{pk+t,pk+t}^{(pk+t-1)},$$
$$b_{pk+t}^{(pk+t)}=b_{pk+t}^{(pk+t-1)}/a_{pk+t,pk+t}^{(pk+t-1)}. \qquad (2.5)$$

［処理Ｂ］．第ｉ行の各要素に対して，次の計算を行う．但し，(p+1)k+1≦ｉ，ｊ≦ｎとする．

$$Reg_i^{(0)}=a_{i,pk+1}^{(pk)},$$
$$Reg_i^{(1)}=a_{i,pk+2}^{(pk)}-Reg_i^{(0)}a_{pk+1,pk+2}^{(pk+1)},$$
$$\cdots,$$
$$Reg_i^{(k-1)}=a_{i,(p+1)k}^{(pk)}-\sum_{m=1}^{k-1}Reg_i^{(m-1)}a_{pk+m,(p+1)k}^{(pk+m)} \qquad (2.6)$$

$$a_{i,j}^{((p+1)k)}=a_{i,j}^{(pk)}-\sum_{m=1}^{k}Reg_i^{(m-1)}a_{pk+m,j}^{(pk+m)}. \qquad (2.7)$$

$$b_i^{((p+1)k)}=b_i^{(pk)}-\sum_{m=1}^{k}Reg_i^{(m-1)}b_{pk+m}^{(pk+m)}. \qquad (2.8)$$

［main操作Ｇ］．n-[n/k]k=0ならば，［処理A_1］～［処理A_k］と［処理Ｂ］までの一連の処理をｐ＝０からｐ＝[n/k]-2まで繰り返し，更に，ｐ＝[n/k]-1として，［処理A_1］～［処理

A_k］までの処理を行う．n-[n/k]k>0ならば，［処理A_1］～［処理A_k］と［処理B］までの一連の処理をp＝0からp＝[n/k]-1まで繰り返し，更に，p＝[n/k]として，［処理A_1］～［処理$A_{n-[n/k]k}$］までの処理を行う．但し，[x]はxを越えない最大の整数を表す．

以上が，多枢軸同時消去計算に基づくガウス消去法の前進消去部分のアルゴリズムである．アルゴリズム上の注意としては，式(2.1)，(2.5)の除数の逆数や変数*Reg*は，レジスタ変数として扱い，レジスタ上に保持するようにする．これは，メモリへのロード回数を減らすためで，これが本アルゴリズムの特徴でもある．枢軸選択方法は，部分枢軸選択とし，列交換の方法で行わなければならない．これは，［処理A_1］～［処理A_k］で，部分的に消去は行われているが，全体での消去（更新計算）は行われていないためである．

2．2　ガウス・ヨルダン法

［処理A_1］～［処理A_k］は，ガウス消去法と同じ．［処理B］は，次の［処理B'］に置き換わる．

［処理B'］．第i行の各要素に対して，式(2.6)，(2.7)，(2.8)を行う．但し，$1 \leqq i \leqq pk$，$(p+1)k+1 \leqq i \leqq n$，$(p+1)k+1 \leqq i \leqq n$とする．

更に，次の［処理C_t］(t=1, 2, ･･･, k-1)が必要である．

［処理C_t］．係数行列の第pk+1行から第pk+t行の各要素及び，既知数ベクトルbの第pk+1成分から第pk+t成分に対して次の計算を行う．但し，$pk+t+2 \leqq j \leqq n$とする．

$$Reg^{(0)} = a_{pk+t,\,pk+t+1},$$
$$Reg^{(1)} = a_{pk+t-1,\,pk+t+1},$$
$$\cdots,$$
$$Reg^{(t-1)} = a_{pk+1,\,pk+t+1}.$$

(2.9)

$$a_{pk+t,\,j}^{(1)} = a_{pk+t,\,j} - Reg^{(0)}\, a_{pk+t+1,\,j}^{(1)},$$
$$a_{pk+t-1,\,j}^{(1)} = a_{pk+t-1,\,j} - Reg^{(1)}\, a_{pk+t+1,\,j}^{(1)},$$
$$\cdots,$$
$$a_{pk+1,\,j}^{(t-1)} = a_{pk+1,\,j} - Reg^{(t-1)}\, a_{pk+t+1,\,j}^{(t-1)}.$$

(2.10)

$$b_{pk+t}^{(t+1)} = b_{pk+t} - Reg^{(0)}\, b_{pk+t+1}^{(1)},$$
$$b_{pk+t-1}^{(t+1)} = b_{pk+t-1} - Reg^{(1)}\, b_{pk+t+1}^{(1)},$$
$$\cdots,$$
$$b_{pk+1}^{(t+1)} = b_{pk+1} - Reg^{(t-1)}\, b_{pk+t+1}^{(1)}.$$

(2.11)

［main操作J］．n-[n/k]k=0ならば，［処理A_1］～［処理A］と［処理B］，［処理C_1］～［処理C_{k-1}］までの一連の処理をp＝0からp＝[n/k]-1まで繰り返し行う．n-[n/k]k>0ならば，［処理A_1］～［処理A_k］と［処理B］，［処理C_1］～［処理C_{k-1}］までの一連の処理をp＝0からp＝[n/k]-1まで繰り返し，更に，p＝[n/k]として，［処理A_1］～［処理$A_{n-[n/k]k}$］までの処理を行い，枢軸数をn-[n/k]kとした［処理B'］を行い，最後に，［処理C_1］～［処理$C_{n-[n/k]k}$］までの処理を行う．

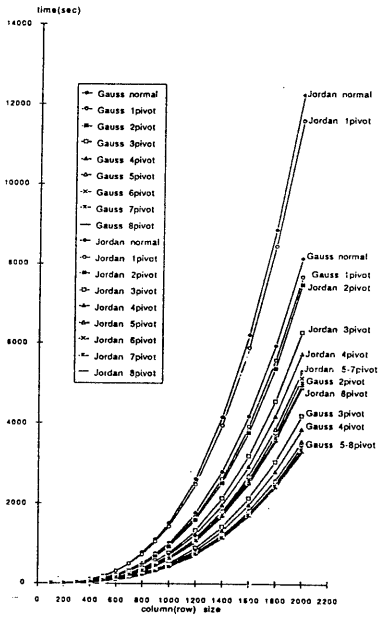
以上が，多枢軸同時消去計算に基づくガウス・ヨルダン法のアルゴリズムである．アルゴリズム上の注意は，ガウス消去法のときと同様．

2．3　逐次処理による評価実験

前節のガウス消去法，ガウス・ヨルダン法（1～8枢軸）ついて，100次元から2000次元の問題に対し，スカラー計算機（単一プロセッサ）による評価を行った．使用したマシンはMIPS RC3260で，CPUがMIPS社製のR3000(25MHz)，FPUがR3010，命令キャッシュが64KB，データキャッシュが64KB，メインメモリが64MBで構成されている．プログラムコンパイル時のオプティマイズレベルは0で，枢軸選択は行っていない．浮動小数点演算は倍精度で行い，処理時間は，消去を始めてから解が求まるまでのCPU時間を測定した．実験結果のグラフを図1に示す．8枢軸の場合，通常の手法の約2．5倍高速である．

参考までに，作成したプログラムの浮動小数点演算の計算量について付記する．枢軸数とは無関係に，ガウス消去法では，乗算は$n^3/3+n^2-n/3$，加減算は$n^3/3+n^2/2-5n/6$，除算はnである．

ガウス・ヨルダン法では，乗算は$n^3/2+n^2/2$，加減算は$n^3/2-n/2$，除算はnである．



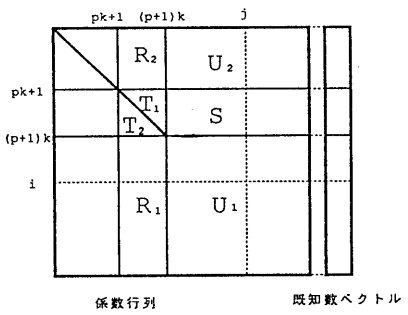
（図１）処理時間

3　多枢軸列同時消去法の並列化手法

ここでは，分散共有メモリ型並列計算機を対象として，前章のアルゴリズムの並列化手法について考える．

3．1　データ割付

図２は係数行列と既知数ベクトルを模式的に示したものである．



（図２）係数行列と既知数ベクトル

この図を基に，係数行列の各分散メモリへのデータ割付方法について考える．係数行列の割付方向としては，①行方向，②列方向，③２次元方向の３通りが考えられる．これらについて，Ⅰ）並列化可能度，Ⅱ）メモリへのロード，Ⅲ）枢軸選択の観点から考えてみる．

①の行方向の場合：

Ⅰ）．図２のT_2部分を消去したときのT_1，S部分の更新計算は並列化できないが，U_1，U_2部分の更新計算やR_1，R_2部分の更新計算は並列化できる．

Ⅱ）．メモリへのロードの減少のためには，１行分のR_1またはR_2部分のｋ個の要素をレジスタ上に保持する必要があり，これは可能である．

Ⅲ）．枢軸選択は，行方向への検索後，列交換の方法で行わなければならないが，検索のための１行分のデータは１つのメモリに保持されているので，検索のためのデータ通信は起こらず，列交換も各メモリ内で行うことができる．

②の列方向の場合：

Ⅰ）．T_1，S部分の更新計算やU_1，U_2部分の更新計算は並列化できるが，R_1，R_2部分の更新計算は並列化できない．

Ⅱ）．メモリへのロードの減少のためには，R_1，R_2部分の要素を全てレジスタ上に保持する必要があるが，これは不可能である．

Ⅲ）．検索のためには，各メモリに分散されている１行分のデータ全てに対し，絶対値の大小比較を行わなければならず，また，列交換はメモリ間で行わなければならない．

③の２次元方向の場合：

Ⅰ）．T_1，S部分の更新計算やU_1，U_2部分の更新計算，R_1，R_2部分の更新計算が並列化できる．

Ⅱ）．上述の②－Ⅱ）と同様．

Ⅲ）．上述の②－Ⅲ）と同様．

以上より，係数行列の割付方向としては，①の行方向が，最善であると言える．なお，①－Ⅰ）で並列化できなかった部分は，クラスタ化した並列計算機では，クラスタ内のＰＥ数での並列

化が可能である．

次に，データの割付方法であるが，枢軸数行未満の単位で割り付けた場合，図２のT_2部分を消去したときのT_1，S部分の更新計算を行う際に，ノード間での通信が必要になるが，枢軸数の整数倍行を単位として割り付ければ通信は必要なくなる．この部分の処理は，U_1，U_2部分の更新計算の前処理の一部で，なるべく速やかに処理されなければならない．また，処理としての一様性を保つ面でも，枢軸数の整数倍行を単位として考えるのが妥当と言える．一方，メモリ当たりのデータ量の均等性を考慮すれば，割り付ける行数の単位は小さい方が良い．従って，枢軸数行づつ割り付けるべきであるということになる．また，クラスタ化した並列計算機の場合，クラスタ内のＰＥ数と枢軸数との関係は，ＰＥ数が枢軸数の約数個であることが望ましい．これは，クラスタ内のＰＥが処理する行数を均等にできるからである．

３．２　処理方法

まず，クラスタ化していない並列計算機の場合について考える．同時消去計算を行う枢軸数はｋとし，係数行列のpk+1行から(p+1)k行を保持しているメモリを含むノードをノードα_pとする．但し，ガウス消去法に関しては前進消去部分についてのみ述べる．

［１］．ノードα_p内のＰＥで，［処理A_1］の処理を行い，その間，他のノードのＰＥでは，［処理Ｂ］の式(2.6)の第1式の計算を行う．ノードα_pに，pk+1行から(p+1)k行以外の行も割り付けられているときは，ノードα_pでも式(2.6)の第１式の計算を行う．ノードα_pでの式(2.1)の計算結果を他のノードのメモリにデータ転送する．

［２］．ノードα_p内のＰＥで，［処理A_2］の処理を行い，その間，他のノードのＰＥでは，［処理Ｂ］の式(2.6)の第２式の計算を行う．ノードα_pに，pk+1行から(p+1)k行以外の行も割り付けられているときは，ノードα_pでも式(2.6)の第２式の計算を行う．ノードα_pでの式(2.5)の計算結果(t=2)を他のノードのメモリにデータ転送する．

，・・・，

［ｋ］．ノードα_p内のＰＥで，［処理A_k］の処理を行い，その間，他のノードのＰＥでは，［処理Ｂ］の式(2.6)の第ｋ－１式の計算を行う．ノードα_pに，第pk+1行から第(p+1)k行以外の行も割り付けられているときは，ノードα_pでも式(2.6)の第ｋ－１式の計算を行う．ノードα_pでの式(2.5)の計算結果(t=k)を他のノードのメモリにデータ転送する．

［k+1］．全ノード内のＰＥで式(2.7)，(2.8)を一斉に行う．但し，ガウス消去法の場合は，第(p+1)k+1行から第ｎ行に対して行い，ガウスヨルダン法の場合は，第１行から第pk行と第(p+1)k+1行から第ｎ行に対して行う．

［k+2］．ノードα_pで［処理C_1］～［処理C_{k-1}］の一連の処理を行う．

［main操作］．n-[n/k]k=0ならば，ガウス消去法の場合は［１］～［k+1］をｐ＝０からｐ＝[n/k]-2まで繰り返し，更にｐ＝[n/k]-1として，［１］～［ｋ］を行う．ガウス・ヨルダン法の場合は［１］～［k+2］までの一連の処理をｐ＝０からｐ＝[n/k]-1まで繰り返し行う．n-[n/k]k>0ならば，ガウス消去法の場合は［１］～［k+1］をｐ＝０からｐ＝[n/k]-1まで繰り返し，更にｐ＝[n/k]として，［１］～［n-[n/k]k］を行う．ガウス・ヨルダン法の場合は，［１］～［k+2］をｐ＝０からｐ＝[n/k]-1まで繰り返し，更に，ｐ＝[n/k]として，［１］～［n-[n/k]k］までの処理を行い，枢軸数をn-[n/k]kとした［k+1］，［k+2］を行う．

次に，クラスタ化した並列計算機の場合について考える．この場合の大きな違いは，操作［１］～［ｋ］と［k+2］が，クラスタ内のＰＥ数で並列化できることである．係数行列のpk+1行から(p+1)k行を保持しているメモリを含むものをクラスタα_pとし，クラスタ内のＰＥ数は，ここでは同時消去行う枢軸数と同じｋとし，そ

れを$PE_1, \cdots, PE_k$とし，PE_m（$1 \leqq m \leqq k$）は，第pk+m行を担当する．

［CL1］．クラスタα_pのPEで，［処理A_1］を行う．その際，PE_m（$1 \leqq m \leqq k$）は第$(p+\nu)k+m$列（$0 \leqq \nu \leqq (n-m)/k-p$の整数）を担当し，各PEで一斉に行う．その間他のクラスタのPEでは，自分の担当行に対して，［処理B］の式(2.6)の第1式の計算を一斉に行う．クラスタα_pに，pk+1行から(p+1)k行以外の行も割り付けられているときは，クラスタα_pでも式(2.6)の第1式の計算を行う．クラスタα_pでの式(2.1)を計算結果を他のクラスタのメモリにデータ転送する．

［CL2］．クラスタα_pのPEで，［処理A_2］の処理を行う．その際，PE_mは第$(p+\nu)k+m$列を担当し，各PEで一斉に行う．その間他のクラスタのPEでは，自分の担当行に対して，［処理B］の式(2.6)の第2式の計算を一斉に行う．クラスタα_pに，pk+1行から(p+1)k行以外の行も割り付けられているときは，クラスタα_pでも式(2.6)の第2式の計算を行う．クラスタα_pでの式(2.5)の計算結果(t=2)を他のノードのメモリにデータ転送する．

，・・・，

［CLk］．クラスタα_pのPEで，［処理A_k］の処理を行う．その際，PE_mは第$(p+\nu)k+m$列を担当し，各PEで一斉に行う．その間他のノードのPEでは，自分の担当行に対して，［処理B］の式(2.6)の第k－1式の計算を一斉に行う．クラスタα_pに，pk+1行から(p+1)k行以外の行も割り付けられているときは，クラスタα_pでも式(2.6)の第k－1式の計算を行う．クラスタα_pでの式(2.5)の計算結果(t=k)を他のノードのメモリにデータ転送する．

［CLk+1］．全クラスタ内のPEで自分の担当行に対して，［処理B］の式(2.7)，(2.8)を一斉に行う．

［CLk+2］．クラスタα_pで［処理C_1］～［処理C_{k-1}］の一連の処理を行う．その際，PE_mは第$(p+\nu)k+m$列を担当し，各PEで一斉に行う．

［main操作］クラスタ化していない並列計算機の場合と同様．

最後に，枢軸選択方法について述べる．担当ノード（クラスタ）のPEで行方向へ検索後，検索した列番号を他のノード（クラスタ）へブロードキャストする．そのデータを基に，直接もしくはインデックスの交換を列交換の方法で一斉に行う．未知数ベクトルに関しては，担当ノード（クラスタ）間でその交換を行う．

3．3 評価

T(k)はk枢軸のときの単一プロセッサでの処理時間，$T_p(k)$はそのときの全プロセッサでの並列処理部の処理時間，$T_s(k)$は逐次処理部の処理時間とする（従って，$T(k)=T_s(k)+T_p(k)$）．更に，$T_{pc}(k)$はクラスタ内部での並列処理部での処理時間，Pは全プロセッサ数，P_cはクラスタ当たりのプロセッサ数である．T_{tr}は転送時間等の並列化によるオーバヘッドとする．処理時間を浮動小数点演算とロード・ストアの和と考え，枢軸選択については考慮せず，並列化によるオーバヘッドは転送のみとし，転送方法はブロードキャストとし，演算，ロード・ストア，転送のサイクル比を，（加減算）：（乗算）：（除算）：（転送）：（ロード・ストア）＝1：1：9：8：10として計算すると，ガウス消去法（前進消去のみ）の場合，

$$T(k)=(12k+20)n^3/3k+31n^2/2 +(20k^2+183k-120)n/6k, \quad (3.1)$$

$$T_p(k)=(12k+20)n^3/3k-(3k^2+2k-10)n^2/k -(3k^2+14k+30)n/3, \quad (3.2)$$

$$T_s(k)=(6k^2+35k-20)n^2/2k +(2k^3+16k^2+81k-40)n/2k, \quad (3.3)$$

$$T_{pc}(k)=(6k^2+35k-20)n^2/2k -(6k^3-54k^2-135k+120)n/6k, \quad (3.4)$$

$$T_{tr}=4n^2-4n. \quad (3.5)$$

ガウス・ヨルダン法の場合，

$$T(k)=(6k+10)n^3/k+(10k^2+11k+20)n^2/2k -(10k^3-90k^2-61k+120)n/6k, \quad (3.6)$$

$$T_p(k)=(6k+10)n^3/k-(6k^2+4k-20)n^2/k -(6k+20)n, \quad (3.7)$$

$$T_s(k)=(22k^2+19k-20)n^2/2k -(10k^3-126k-181k+120)n/6k, \quad (3.8)$$

$$T_{pc}(k)=(22k^2+19k-20)n^2/2k -(32k^3-132k^2-263k+240)n/6k, \quad (3.9)$$

$$T_{tr}=4(P_c+1)n^2-4(P_c+1)n. \quad (3.10)$$

一方，クラスタ化していない並列計算機での処理時間Tnp(k)は，

$$Tnp(k)=T_p(k)/P+T_s(k)+T_{tr}. \quad (3.11)$$

クラスタ化した並列計算機での処理時間Tcl(k)は，

$$Tcl(k)=T_p(k)/P+T_{pc}(k)/P_c+T_s(k)-T_{pc}(k) +T_{tr}. \quad (3.12)$$

従って，式(3.1)～(3.10)を式(3.11)，(3.12)に代入すると，ガウス消去法の場合，

$$Tnp(k)=(12k+20)n^3/3Pk+\{6(P-1)k^2+(43P-4)k -20(P-1)\}n^2/2Pk-\{6(P-1)k^3+(48P-28)k^2 +(219P-60)k-120P\}n/6Pk, \quad (3.13)$$

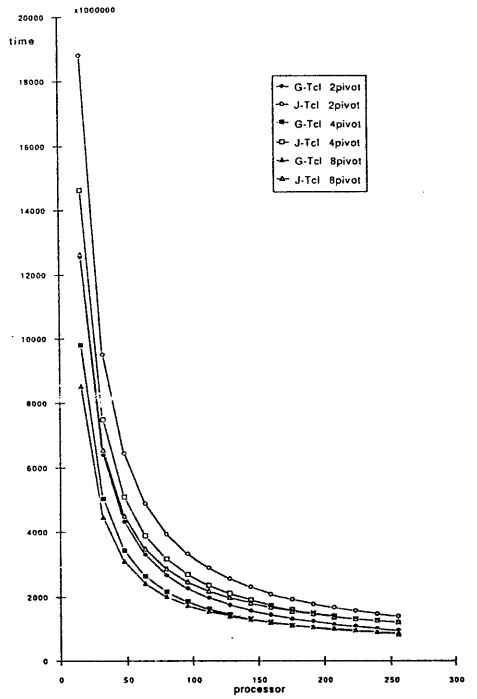
$$Tcl(k)=(12k+20)n^3/3Pk+[6(P-P_c)k^2 +\{8PP_c(P_c+1)+35P-4P_c\}k+20(P_c-P)]n^2/2PP_ck +\{6(2PP_c-P-P_c)k^3+(54P-28P_c-6PP_c)k^2+(135P -60P_c+132PP_c+24PP_c^2)k-120P\}n/6PP_ck, \quad (3.14)$$

ガウス・ヨルダン法の場合，

$$Tnp(k)=(6k+10)n^3/Pk+\{(22P-12)k^2+(27P-8)k -20(P-2)\}n^2/2Pk-\{10Pk^3-(126P-36)k^2 -(157P-120)k+120P\}n/6Pk, \quad (3.15)$$

$$Tcl(k)=(6k+10)n^3/Pk+[(22P-12P_c)k^2 +\{8PP_c(P_c+1)+19P-8P_c\}k+20(2P_c-P)]n^2/2PP_ck +\{(22PP_c-32P)k^3+(132P-36P_c-6PP_c)k^2+(263P -120P_c-106PP_c-24PP_c^2)k-240P+120PP_c\}n/6PP_ck. \quad (3.16)$$

TnpとTclを比較すると，nの３次の項の係数は等しいので，nの２次の項の小さい方が処理時間は速い．従って，処理方法，枢軸数，プロセッサ数が同一ならば，クラスタ化した並列計算機の方が高速である．Tcl(Tnp)について考察する．k<<nと仮定できるので，もしP<<nならば，式(3.13)～(3.16)におけるnの３次の項が関数値に大きく影響し，そのnの３次の項の係数がkに関して単調減少であることから，Tcl(Tnp)は，kに関して単調減少となる（P, P_c, nは固定）．n=3000, k=2, 4, 8のときのTclのグラフを図３に示す．図中，Gはガウス消去法，Jはガウス・ヨルダン法を表す．



（図３）Tclのグラフ

図３において，プロセッサ数が増えたところで

は，nの２次の項の影響が大きくなるので，枢軸数の多い方が遅くなっている部分が見られる．

次に，単一プロセッサ処理に対する加速率について考察する．加速率を次式で定義する．

$$(\text{加速率NP}) = T(k) / Tnp(k) \qquad (3.17)$$

$$(\text{加速率CL}) = T(k) / Tcl(k) \qquad (3.18)$$

この定義に基づき，k=2,4,8;n=3000,10000,P_c=kの場合について調べた結果を図４，５に示す．図中，NPはクラスタ化していない並列計算機の場合，CLはクラスタ化した並列計算機の場合を示す．図から，P,nを固定すると，枢軸数が増えるにつれ加速率は悪くなることが解る．

4　おわりに

本手法は，分散共有メモリ型並列計算機を対象としたものだが，理論値による評価から，クラスタ化した方が，処理時間，加速率の点で有利であることが解る．また，問題規模がプロセッサ数に対して大きいときは，枢軸数を増やすことが処理の高速化につながる．この仮定が成り立たないときは，問題規模とプロセッサ数に応じて枢軸数を可変的に使い分ける必要がある．

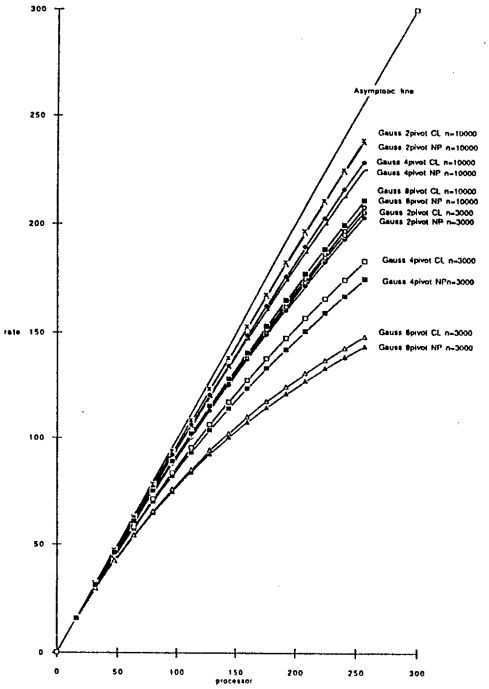
以上，本報告書を通じ，本手法の有効性を示せたものと思う．今後は，クラスタ化した並列計算機における枢軸数とクラスタ内のプロセッサ数との最適関係についての考察と実際の並列計算機による評価を行いたい．

参考文献

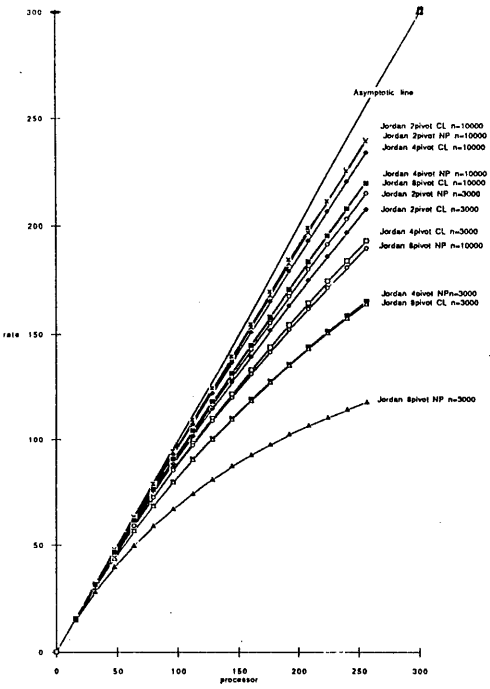
1)Armstrong, J.:ALGORITHM AND PERFORMANCE NOTES FOR BLOCK LU FACTORIZATION, International conference on parallel processing, Vol.3, pp.161-164(1988).

2)日本物理学会編：スーパーコンピュータ，p.278，培風館(1985)．

3)村田健郎，小国力，唐木幸比古，スーパーコンピュータ -科学技術計算への適用，p.304，丸善株式会社(1985)．



（図４）ガウス消去法の加速率



（図５）ガウス・ヨルダン法の加速率