# 高速数値計算のための分散処理への対応について

総合情報学部情報科学科
河野敏行

## 1. 課題概要

センター実習室の複数のPCを利用して線形計算を各PCに分散処理をすることによって，大規模な問題を高速に解くことを目標として実験に取り組んだ[1]．現在，コンピュータの高速化が進み，より複雑な問題を取り扱うことが可能となり，より問題が複雑になるに従い，その解析には多くの時間が掛かり，1台の計算機では解くことが困難となっている．昨年度は，その環境を構築するためにJavaScriptとPHPを利用したサーバと複数PCをクライアントとして分散処理するシステムを試作した．数値実験としては3重対角行列に対する反復解法を行った．今年度は，効率良く計算するための線形問題の分割方法について議論する．この分割の仕方によって分散計算による解の精度が大きく変化することがわかった．さらに，数値解析的に反復計算を加速する前処理を分散処理に適用した．

## 2. 反復解法について

自然現象などを表す微分方程式を解く際に得られる次式で示されるような線型方程式を解くことは全体の解析において多くの時間が費やされることが知られている．

$$Ax = b$$

ここで，$A$は$n \times n$行列，$x$は$n$次の未知ベクトル，$b$は$n$次の既知ベクトルである．この係数行列$A$は疎といわれる零要素の多い行列が得られる．このような問題を解くには計算の効率などから反復解法が使われている．反復解法として我々は定常反復法の代表としてGauss-Seidel法を用いることとする．数値実験では行列の要素を特定した3重対角行列および5重対角行列のテスト行列を扱うこととする．ベクトル$b$は真の解$x^* = (1,1,\cdots,1)^T$と設定して，$b = Ax^*$を計算することで与える．初期近似ベクトルは$x^{(0)} = (0,0,\cdots,0)^T$と設定する．今回のテスト問題も前回の研究と同様に以下に示すような行列$A$を扱うこととする．すなわち対角成分をすべて1.0，上下に1ないし2つの要素を持つ3重または5重対角行列とする．ここで，$p,q,r,s$は任意に設定するとする．3重対角行列の場合は$q = s = 0$となる．数値実験において扱う係数行列は理論上取り扱いを容易にするために，$p,q,r,s$を非正とし，$|p|+|q|+|r|+|s|<1$となるように設定することとする．すなわち，係数行列$A$が狭義優対角なZ行列となるようにする．

$$A=\begin{bmatrix} 1.0 & p & q & 0 & \cdots & 0 \\ r & 1.0 & p & q & \ddots & \vdots \\ s & r & 1 & p & \ddots & 0 \\ 0 & s & r & 1 & \ddots & q \\ \vdots & \ddots & \ddots & \ddots & \ddots & p \\ 0 & \cdots & 0 & s & r & 1 \end{bmatrix}$$

次に反復解法の加速として前処理を用いるが，我々はこれまでに次のように元の線型方程式に正則行列$P$を前処理行列として左から乗算した式を前処理化線型方程式として定義し，この方程式に対して通常の定常反復法を適用することとする．

$$PAx = Pb$$

前処理行列$P$としては係数行列$A$の要素を用い容易に得られる行列とする．1991 年に A.D.Gunawardena らによって$P = I + S$という前処理が提案された[2]．ここで，$S$は$A = (a_{ij})$に対して

$$S=\begin{bmatrix} 0 & -a_{12} & 0 & 0 \\ & & \ddots & 0 \\ & & & -a_{n-1n} \\ 0 & & & 0 \end{bmatrix}$$

とあらわされる，すなわち係数行列の対角のひとつ右の要素にマイナスを乗じた要素で示される．このような前処理を用いた前処理付反復法は，与えられた係数行列が疎である場合は，前処理化係数行列の導出とそのために生じる非零要素の増大という問題が生じる．その問題の回避として，前処理化アルゴリズムを提案した[3]．$P = I + S$を用いた前処理付 Gauss-Seidel 反復アルゴリズムをアルゴリズム 1 に示す．

アルゴリズム 1

Do　k = 0, maxiterate

　Do　i = 1, n

$$r_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)})/a_{ii} - x_i^{(k)}$$

$$x_i^{(k+1)} = x_i^{(k)} + \omega\left( r_i^{(k+1)} - \frac{a_{ii+1}}{a_{ii}} r_{i+1}^{(k)} \right)$$

　enddo

　if x accurate enough then iteration quit

enddo

ここで，$k$は反復回数を示し，最大反復回数 maxiterate 回まで繰り返す，または，近似解$x$が解に近づき収束条件を満たした場合は反復を終了する．そしてベクトル$r$の各行の要素は1回の反復で得られる残差ベクトルの要素であるが，最後の n 行目に対しては，$r_{n+1}$は存在しないため 0 とする．そして，その残差ベクトルに前処理要素$s_{ii+1} = -a_{ii+1}$（ただし，対角要素が 1.0 でない場合を考慮し，$a_{ii}$で除算している．しかしながら，今回のように対角要素をすべて 1 と設定した場合は必要ない）を乗じて加えることで，前処理効果を与えている．そして$\omega$は SOR 法のパラメータである．実験では，$\omega = 1$の場合の Gauss-Seidel 法のみを利用する．このアルゴリズムを分散処理に利用して，反復回数を減少させることとする．分散処理への対応については 4 節で説明する．

まずは，Gauss-Seidel 反復法を分散処理する場合の問題の解決について議論する．

## 3.　Gauss-Seidel 反復法の分散処理への対応

Gauss-Seidel 反復法を分散処理への対応について考える．Gauss-Seidel 法は，k 行目の計算に 1 から k-1 行目の計算で得られた最新データを使う計算手法であるが，分散をした場合は正しく最新データを使うことはできなくなる．そのためにいくつかの工夫を行う必要がある．今回は，行列を分割した際に生じる誤差を小さくするために行列の重複を行うことについて議論する．

まず，PC クラスタによる分散処理を行うために係数行列$A$を PC クラスタの数によって行を分ける．各 PC クラスタの有効となる行列の次数は分散する PC クラスタの数で割ることとし，端数は最後の PC クラスタへの負担とすることとする．すなわち，行列の次数が 100 に対して PC クラスタの数が 3 台である場合は，最初の 2 台は 33 行分で，最後の 1 台は 34 行分を扱うこととする．一般に，分散処理に対しては Jacobi 反復法が適しているといわれている．なぜなら，計算は各行で分断されており，1 行ごとの更新データを 1 回の反復の中で利用することが要らないからである．しかしながら，数値的な計算速度としては Gauss-Seidel 法の方が優れている．利用するためにはいくつかの工夫が必要となる．ここで，分散の仕方について議論を行うが，正確な計算結果を得るためには，いくつかの重複度をもって計算することが必要となる．ここでいう重複度とは，たとえば，次数 100 に対して，1 番目の PC クラスタは 1-33 行目を計算として担当するが，34 行目のデータを追加して持つ場合，重複度 1 と定義する．このようにした場合，2 番目の PC クラスタは 34-66 行ではなく，33-67 行を計算することとする．そして，全体の近似解ベクトルに対しては，34-66 の値を用いることとする．表 1 に PC クラスタを 3 台とし，次数 100 に対して重複度を変化させた場合の計算する行の総和の変化を示した．そして，図 1 にその計算結果を示す．扱った行列の要素は$p = -0.4, q = 0, r = -0.3, s = 0$として設定した(問題 A)．横軸には計算された反復回数が示され，縦軸には反復ごとの残差ノルムを対数で示した．

表 1　分散 3 において次数 100 の問題に対して重複度による計算行の違い

| PC ＼ 重複度 | | 1 | 6 | 10 | 15 |
|---|---|---|---|---|---|
| PC クラスタ 1 | 開始行 | 1 | 1 | 1 | 1 |
| | 終了行 | 34 | 39 | 43 | 48 |
| PC クラスタ 2 | 開始行 | 33 | 28 | 24 | 19 |
| | 終了行 | 67 | 72 | 76 | 81 |
| PC クラスタ 3 | 開始行 | 66 | 61 | 57 | 52 |
| | 終了行 | 100 | 100 | 100 | 100 |
| 計算する行の総和 | | 104 | 124 | 140 | 160 |

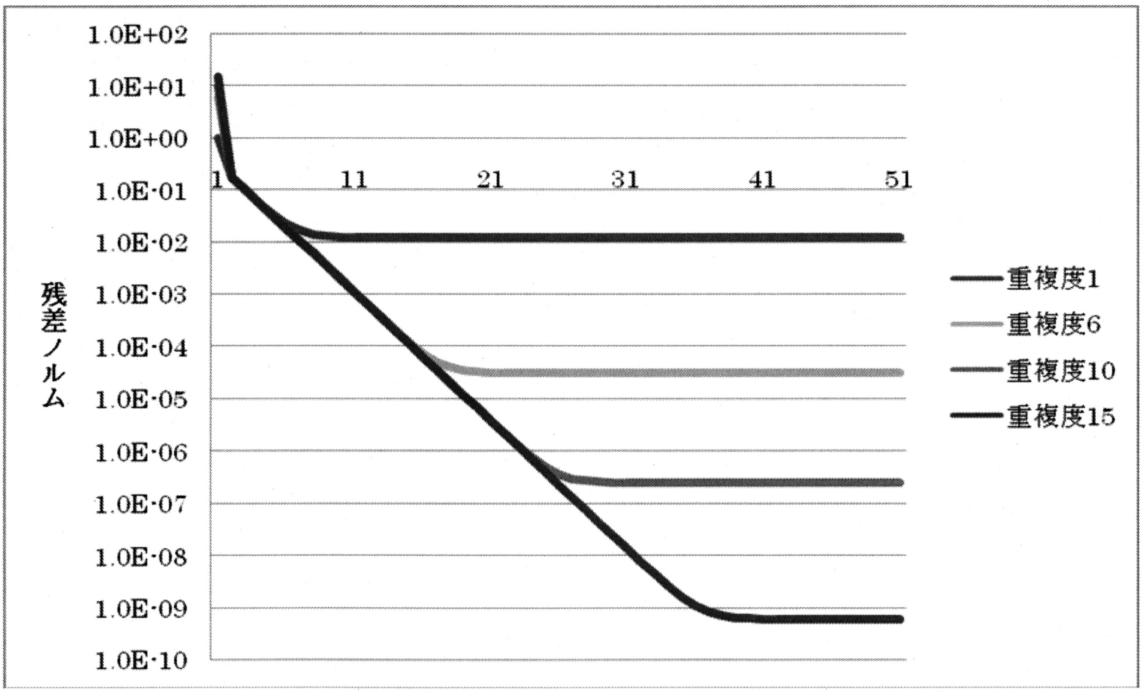


図 1　分散 3 において次数 100 の 3 重対角行列に対して重複度による収束状況の違い

具体的に表 1 での残差ノルムは各反復回における近似解 $x^{(k)}$ を用いて $\dfrac{\left\|b - Ax^{(k)}\right\|_2}{\|b\|_2}$ で示している．図 1 から分かるように重複度が増えるごとに得られる解の精度が変化している．そして収束の速さは同じである．すなわち，必要な解の精度を得るために必要な反復回数は変わらないことが分かる．重複度が 5 以下の場合は重複度 1 の場合と同様な結果が得ら

れ，解の精度は反復を繰り返したとしても２桁の精度しかえらない．これは係数行列をPCクラスタに強制的に分割することによって分割された近辺で誤差が生じていることが計算結果を見ることでわかる．重複度が６以上であれば，４桁程度の精度の解が得られるという結果が得られた．次に，図２に次数300の場合の計算結果を示す．行列の３重対角をなす各要素は問題Ａである．

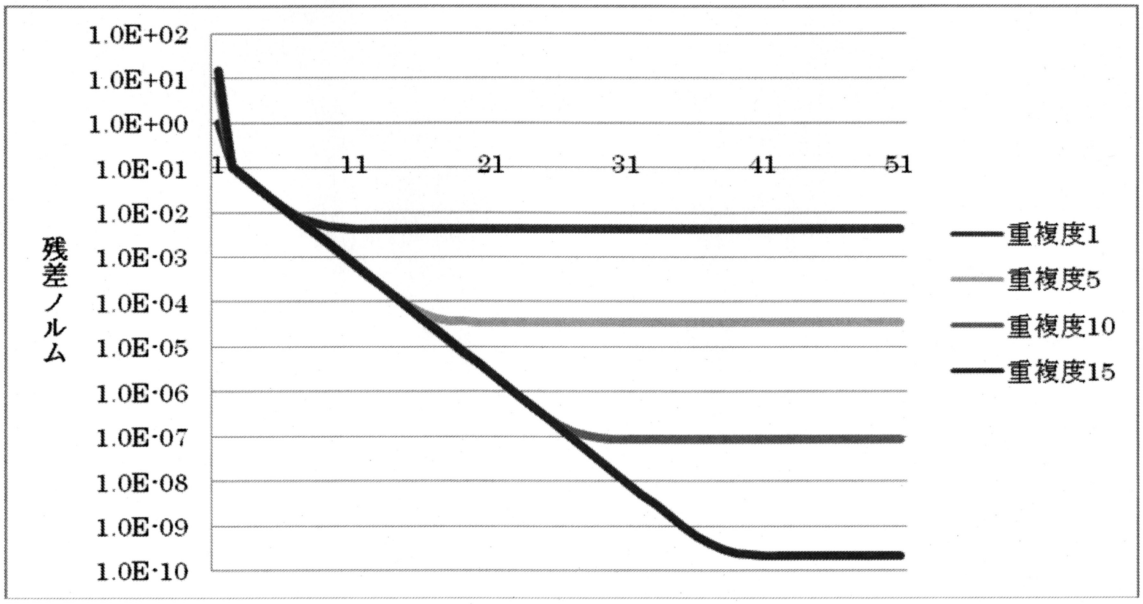


図２　分散３において次数300の３重対角行列に対して重複度による収束状況の違い

図２から分かるように次数が300の場合も重複度による変化は次数が100のときと同様な結果が得られた．

次に重複度を一定にして，次数による変化について調べる．重複度を10として，次数を100，300，500，1000とした結果を図３に示す．図３から分かるように，次数が増加すると解の精度が多少上がっていることが分かる．これは，次数の増加に反して，PCクラスタの数は３と一定であり，行列の分割によって精度を下げている個所は比較的に少なくなるため，この現象が起こると思われる．PCクラスタの数が増加した場合は精度を下げる要因が増えることになるが，少ない次数の問題に対してPCクラスタを増やす必要はないため，この問題は大きな影響とならないであろう．しかしながら，少ない重複度で，精度を上げるための工夫も考えられるが，今後の課題とする．

次に要素を増やし，５重対角行列に対する重複度の変化について議論する．

５重対角にした場合は３重対角の場合よりも問題が複雑となり，より多くの反復回数が必要となる．５重対角行列の要素を $p=-0.2, q=-0.3, r=-0.1, s=-0.2$ とした場合(問題Ｂ)の重複度の違いによる計算結果を図４に示す．

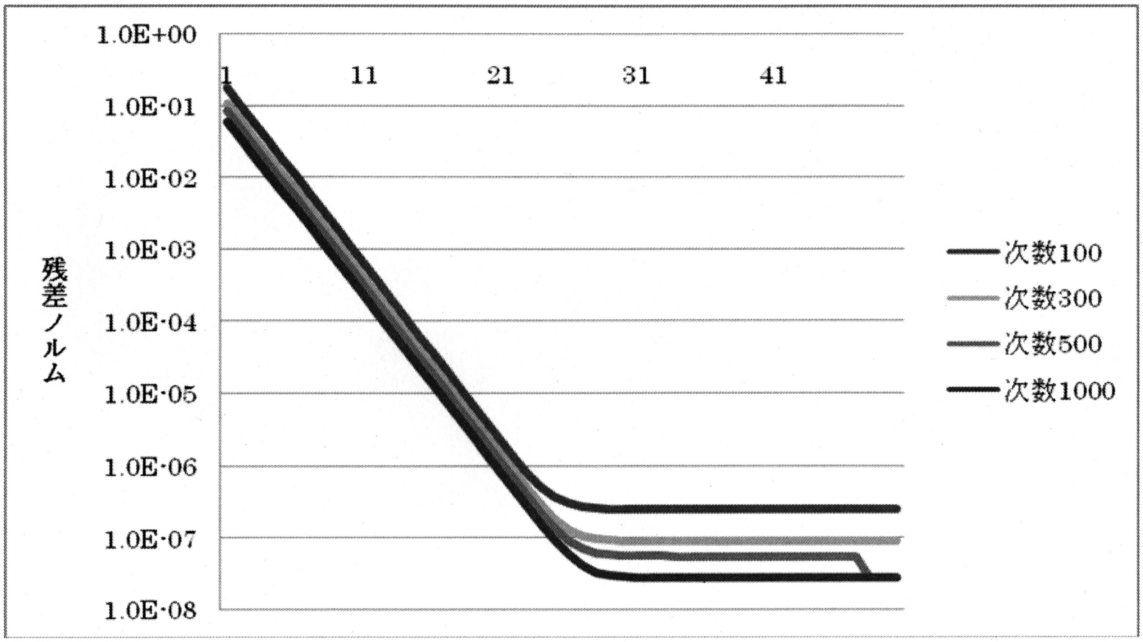


図 3　分散 3 において次数 100,300,500,1000 の 3 重対角行列に対する収束状況の違い

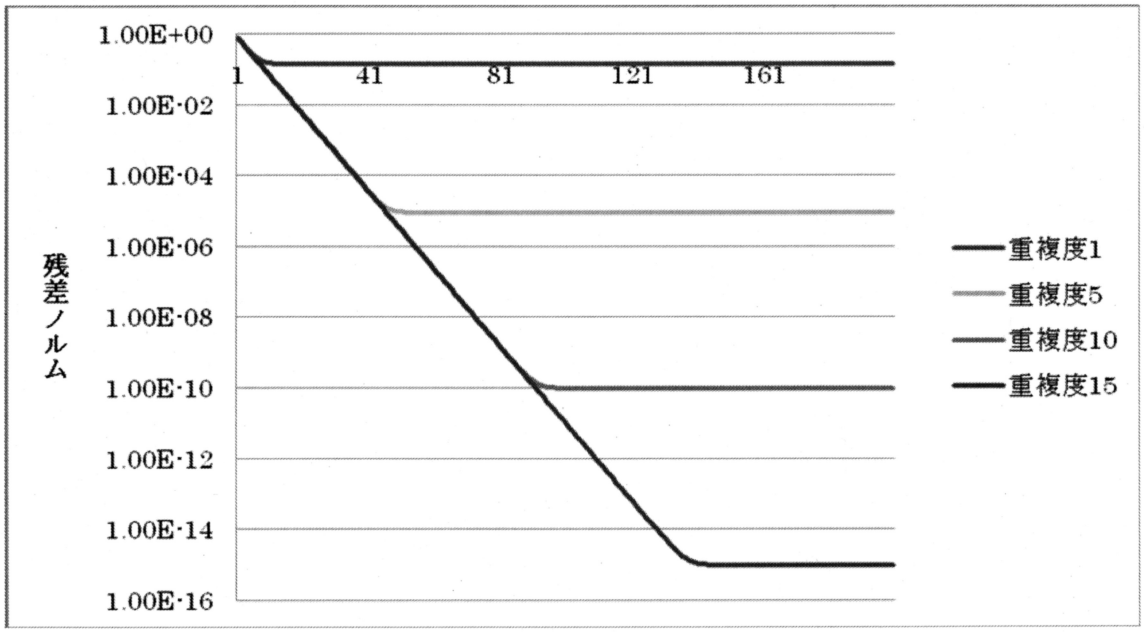


図 4　分散 3 において次数 300 の 5 重対角行列に対して重複度による収束状況の違い

図 4 では，解の精度を得るために反復回数を 200 として計算した．重複度によって，3 重対角と同様の結果が得られた．

次に前処理の分散化について議論する．

## 4.　前処理の分散処理への対応

先の 1000 次の問題 A に対して 1 CPU で処理をした場合の結果を図 5 に示す．前処理を施した場合，計算回数が小となることはこれまでの研究でわかっている[1,3].

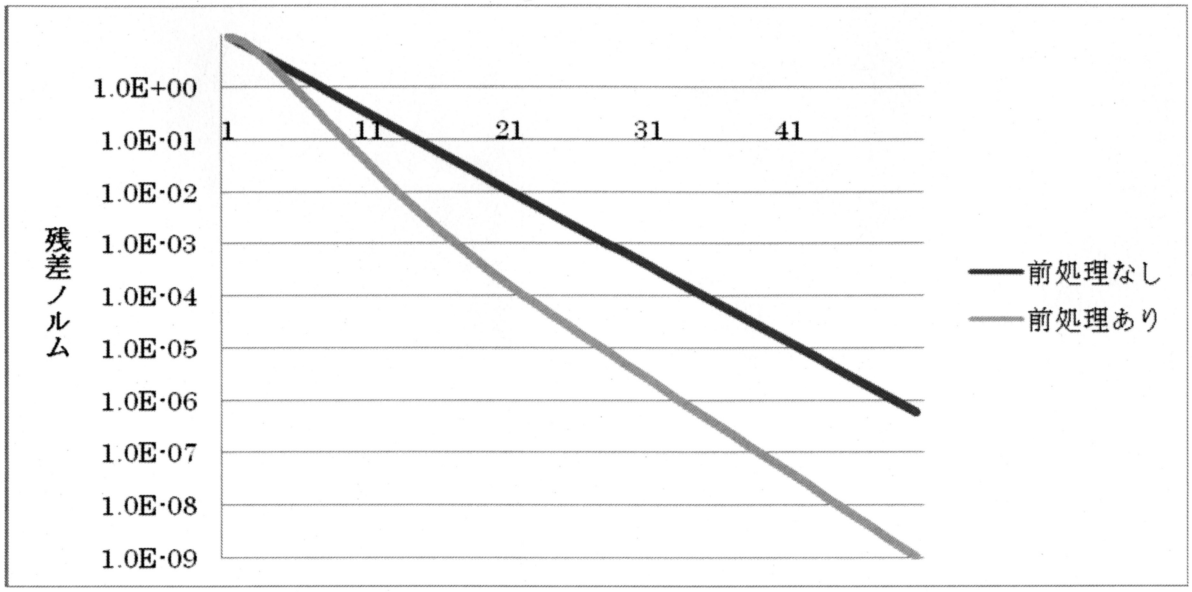


図 5　分散なしの次数 1000 に対する前処理効果

次に，前処理アルゴリズムを分散に対応したものをアルゴリズム 2 に示す．アルゴリズム 1 とほぼ同様に定義することができる．

アルゴリズム 2

Do　k = 0, maxiterate

(各クライアントで分割　ここから)

Do　i = 1, last row of eachPC

$$r_i^{(k+1)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)})/a_{ii} - x_i^{(k)}$$

(最後の行以外)　$$x_i^{(k+1)} = x_i^{(k)} + \omega\left(r_i^{(k+1)} - \frac{a_{ii+1}}{a_{ii}} r_{i+1}^{(k)}\right)$$

(最後の行のみ)　$$x_i^{(k+1)} = x_i^{(k)} + \omega r_i^{(k+1)}$$

enddo

(分割ここまで)

if x accurate enough then iteration quit

enddo

アルゴリズム 2 において，前処理の効果は行列の分割された最後の行に適用されないことが分かる．PC クライアントを 3 台にした場合の計算結果を図 6 に示す．次数は 1000 の問題 A に対して重複度は 10 としている．

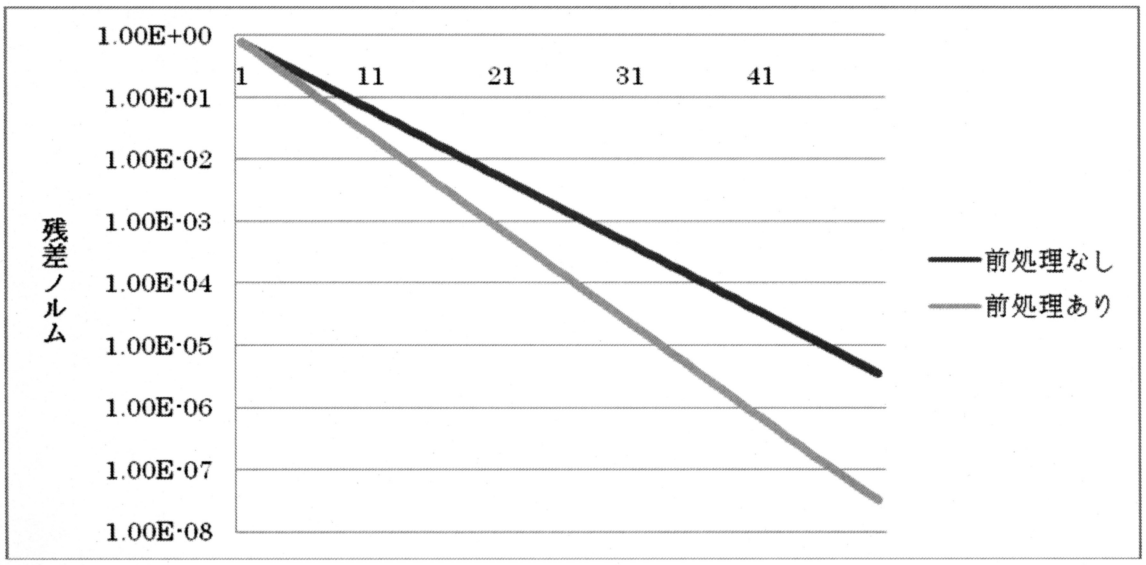


図 6　分散 3 における次数 1000 に対する前処理効果の比較（重複度 10）

図 6 から分かるように前処理の効果によって必要な解の精度を得るための反復回数が減少していることが分かる．分散なしの場合と同様に分散を行った場合に対しても前処理の効果が得られた．

## 5.　考察

今回の実験において，昨年度作成した分散計算環境[1]を利用して，数値計算を行う際に生じる分散の処理の仕方，前処理の分散について研究を行った．分散の処理の仕方としては，線型方程式を分割する際に生じる誤差を減少させるために，重複する部分を増加させることで対応でき，解の精度を上げることが分かった．この実験の結果，重複度 10 程度あれば，必要な解の精度が得られるであろうと思われる．次数が増えることによって，この重複度による計算量の増加の割合は減少し，計算の負担になるとは限らないと思われる．前処理の効果は各 PC クライアントの最後の行にだけ効果が得られないが，1 クライアントの時と同様に効果が得られた．さらにいろいろな前処理法を用いることが考えられる．

今後の課題として重複度を増やさずに，補正式を用いることについて改良することが残されている．そして，昨年度作成したシステムを利用し，実際に計算を行うことが課題であり，計算問題としてより具体的な問題を扱うことが今後の課題である．

## 参考文献

[1] 河野敏行，高速数値計算の実験，岡山理科大学情報処理センター研究報告，第 28 号 (2007)1-6

[2] A.D.Gunawardena, S.K.Jain and L.Snyder, Modified Iterative Methods for Consistent Linear Systems, Linear algebra Appl., 154-156(1991)123-143.

[3] 河野敏行，適応型前処理付反復法について，日本応用数理学会論文誌, 15,3(2005)235-243.

[4] 小国力訳，J.J.Dongarra, I.S.Duff, D.C.Sorensen, H.A. van der Vorst，コンピュータによる連立一次方程式の解法，丸善株式会社,1993.