ハイパフォーマンスコンピューティング 60－3
（1996. 3. 4）

BiCGStab(ℓ) 法の収束特性について

野口雄一郎　稲津隆敏　野寺　隆†

非対称な連立1次方程式を解く反復解法の1つである BiCGStab 法は，BCG 法の残差ノルムの収束性を滑らかにしたものであり，その中で1次の多項式の最小化を行うものである．近年，この方法を改良した BiCGStab2 法が提案されている．この算法は1次の多項式に加え，2次の多項式の最小化を行うことで残差ノルムの収束の安定性を得るものである．BiCGStab(ℓ) 法はこれらの方法を一般化し，高次の多項式の最小化を行う方法である．これら一連の BiCGStab 関連の算法を分散メモリ型高並列計算機 富士通 AP1000 上で実行し，それぞれの方法の残差ノルムの収束性を比較検討する．

The convergence properties of BiCGStab(ℓ) Method

YUUICHIROU NOGUCHI, TAKATOSHI INADU, TAKASHI NODERA†

For solving the large and sparse non-symmetric linear systems of equations BiCGStab method is known as one of the iterative solvers. This method smoothes the residual norm and minimizes one degree polynomials. Recently BiCGStab2 method has been proposed. Minimizing two degree polynomial, this method stabilizes the residual norm for complex eigenvalues. BiCGStab(ℓ) method is generalized by these methods and minimizes higher degree polynomials. These iterative codes are parallelized and conclusions are drawn on the effectiveness of the different schemes based on results obtained from 64 processor AP1000.

1. はじめに

BiCGStab 法[1),5)]は大型の疎行列を係数とする連立1次方程式

$$Ax = b \tag{1.1}$$

の近似解を求めるための反復解法である．ただし，行列 A は大きさ $n \times n$ の正則な非対称行列で，b は n 次元のベクトルとする．

BiCGStab 法では BCG 法で得られる残差ベクトルと1次の多項式を組合せ，これを各反復で最小化する．これにより BCG 法の残差の収束の不規則性を改善し，収束を滑らかにしている．BiCGStab 法の k 回目の反復における残差は

$$r_k = (I - \alpha A)q_{k-1}(A)\tilde{r}_k \tag{1.2}$$

という式で表され，$\|r_k\|_2$ が最小になるように α が選ばれる．ただし，$\tilde{r}_k$ は BCG 法の k 回目の反復における残差ベクトルであり $q_k(x) = (1 - \alpha x)q_{k-1}(x)$ とする．各反復で必要とする行列とベクトルの積の数は BCG 法と同じであり，このアルゴリズムでは A^T の計算は必要としない．BiCGStab 法は多くの問題に対してよい収束性を示している．しかし $\alpha \approx 0$ となる場合，残差ノルムの収束が悪くなることがある．

これを改善するために作られたのが以下に述べる一連の BiCGStab 法である．

2. BiCGStab2 法

BiCGStab2 法[2)]では，BiCGStab 法で用いている1次式に加え 2 次の多項式の最小化を行っている．BiCGStab2 法の k 回目の反復における残差 r_k は

$$\begin{aligned} &(I - \alpha A)q_{k-1}\tilde{r}_k && \text{if } k = 2m \\ &((1-\beta)q_{k-2} + (\beta + \gamma A)q_{k-1})\,\tilde{r}_k && \text{if } k = 2m+1 \end{aligned} \tag{2.1}$$

という式で表される．ただし，変数 α，β，γ はいずれも実数である．奇数回目の残差をこのように決めることで $q_k(0) = 1$ が保証され，β，γ を変化させることで $q_k(0) = 1$ を満たす任意の2次式が選べることになる．

(2.1) 式は次式のように書き変えられる．

† 慶應義塾大学 理工学部
Faculty of Science and Technology, Keio University

$$r_k = (I - \omega A)(I - \tilde{\omega} A) q_{k-2}(A) \tilde{r}_k \qquad (2.2)$$

ただし, ω, $\tilde{\omega}$ は実数, もしくは共役な複素数である. つまり実際の計算は実数でも複素数を扱っていることになる.

この BiCGStab2 法の奇数回目の反復では 2 次式の最小化を行っている. 実際には (2.1) 式を展開して, 次のような式に直してこれを最小化することになる.

$$r_k = (q_{k-2} + \beta(q_{k-1} - q_{k-2}) + \gamma(A q_{k-1}))\tilde{r}_k \qquad (2.3)$$

この式に対して大きさ $n \times 2$ の行列 B_{m+1} を

$$B_{m+1} = [\, q_{k-1} - q_{k-2} \mid A q_{k-1} \,] \qquad (2.4)$$
$$k = 2m + 1$$

のように定めると (2.3) 式の最小二乗解は

$$\begin{pmatrix} \beta \\ \gamma \end{pmatrix} = -(B_{m+1}^T B_{+m+1})^{-1} B_{m+1}^T q_{k-2}$$
$$k = 2m + 1$$

のようにして求められる. $x = q_{k-1} - q_{k-2}$, $y = A q_{k-1}$ としてこの式を書き直すと, 次のようになる.

$$\beta = \frac{(x,x)(x,q_{k-1}) - (x,y)(y,q_{k-1})}{(x,x)(y,y) - (x,y)(x,y)} \qquad (2.5)$$

$$\gamma = \frac{-(x,q_{k-1})(x,y) + (x,x)(y,q_{k-1})}{(x,x)(y,y) - (x,y)(x,y)} \qquad (2.6)$$

このように逆行列を計算するアルゴリズム中には内積の計算が数多く用いられる.

この算法は奇数回目の反復では 2 次式の最小化を行っているが, 偶数回目の反復では BiCGStab 法と同じ 1 次式の最小化を行っている. このため, BiCGStab 法で残差がうまく収束しない場合この 1 次式が原因で BiCGStab2 法の残差の収束も悪くなることがある.

3. BiCGStab(ℓ) 法

BiCGStab(ℓ) 法[3] は BCG 法で得られたベクトルと ℓ 次の多項式を組合せ, これを最小化する算法である. $k = m\ell + \ell$ とすると BiCGStab(ℓ) 法の k 回目の反復での残差は

$$r_k = p_m(A) q_{k-\ell}(A) \tilde{r}_k \qquad (3.1)$$

と表される. ただし, $q_{k-\ell}(A) = p_{m-1} p_{m-2} \cdots p_0$ であり, $p_m(0) = 1$ を満たす. なお, ℓ 次の多項式 p_m は $\|r_k\|_2$ が最小になるように選ばれる.

この算法は BiCGStab2 法を一般化し, 高次の多項式を用いている. BiCGStab2 法と同じ方法でこれを最小化すると算法が複雑になり必要な記憶領域が増えてしまう. そこで BiCGStab(ℓ) 法では計算方法を工夫することで同じ算法で高次の最小化を行えるようにしている.

BiCGStab(ℓ) 法は, BCG 法のベクトル $\tilde{r}_k$ に代えて $k = m\ell + \ell$ ($m = 0, 1, 2, \cdots$) の時に限り r_k, u_k, x_k を計算する. 前の反復で得られたベクトルから r_k を作り出す過程を一つのまとまりと考えると, この過程は大きく二つに分けられる. BCG 法を使う部分と ℓ 次の MR(Minimum Residual) 多項式を用いる部分である. 以下はこの二つの過程を分けて示す.

(1) **BCG 法を使う部分**

前反復で得られた残差ベクトル r_k に対し, BCG 法の反復を ℓ 回適用する. これで得られたベクトルを $\tilde{r}_k$ とすると, 副産物として計算の途中で $A^i \tilde{r}_k (i = 0, \cdots, \ell)$ が得られる. 通常の BCG 法では 1 回の反復で A, A^T の計算を必要とするが, この BCG 法を ℓ ステップ行う BiCGStab(ℓ) 法は $2 \times \ell$ 回の行列 A の計算を必要とするだけである. つまり BiCGStab(ℓ) 法はベクトルと行列の積の量については BCG 法と同じだけの計算量しか必要としない.

(2) **ℓ 次の MR 多項式を用いる部分**

BCG 法を用いた部分で作られた $A^i \tilde{r}_k (i = 0, \cdots, \ell)$ を用いて最小の残差 r_k を見つける. つまり $\|r_k\|_2$ が最小になるように

$$r_k = \tilde{r}_k + a_1 A \tilde{r}_k + a_2 A^2 \tilde{r}_k + \cdots + a_\ell A^\ell \tilde{r}_k$$

の係数 $a_1, a_2, \cdots, a_\ell$ を選ぶことになる. この係数を GMRES(ℓ) 法で最小化すると行列とベクトルのかけ算を ℓ 回必要とする. しかし BiCGStab(ℓ) 法ではグラムシュミットの直交化法を用いることでこの部分では行列とベクトルのかけ算を必要としない.

以下, $A^i \tilde{r}_k (i = 0, \cdots, \ell)$ から r_k を求めるアルゴリズムを示す. ただし $\hat{r}_i = A^i \tilde{r}_k$ とする.

$$R = (\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_\ell) \qquad (3.2)$$

とする. これをグラムシュミットの直交化法で直行化したものを

$$Q = (q_1, q_2 \cdots, q_\ell) \qquad (3.3)$$

とする. また T は $R = Q\,T$ となるような大きさ $\ell \times \ell$ の下三角行列とし, 大きさ $\ell \times \ell$ の行列 D を

$$D = Q^T Q = \mathrm{diag}(\|q_1\|_2, \|q_2\|_2, \cdots, \|q_\ell\|_2)$$

と定める. ここで明らかに行列 D と T は正則であり, Q と R は非正則である. この時

$$\|r_k\|_2 = \|\hat{r}_0 - R\vec{\gamma}\|_2 \qquad (3.4)$$

が最小になるような $\vec{\gamma}$ を求めればよい. (3.4) 式は

$$\|r_k\|_2 = \|\hat{r}_0 - QT\vec{\gamma}\|_2 \qquad (3.5)$$

と書き換えることが出来るので, これを最小化することにする. この $\vec{\gamma}$ は, 最小二乗解として一般に

$$\vec{\gamma} = ((QT)^T QT)^{-1})(QT)^T \hat{r}_0 \qquad (3.6)$$

として与えられる. $D = Q^T Q$ を用いると, (3.6) 式は

$$\vec{\gamma} = T^{-1} D^{-1} Q^T \hat{r}_0 \qquad (3.7)$$

とすることが出来る. よって最小となる残差ベクトル r_k は

$$r_k = \hat{r}_0 - R\vec{\gamma} = \hat{r}_0 - QD^{-1}Q^T \hat{r}_0 \qquad (3.8)$$

となる.

この算法では ℓ 本のベクトルを直交化するようなグラムシュミットの直交化を行う. そのため ℓ が大きくなると直交化の部分の計算量が $O(\ell^2)$ で増えていく. 計算時間を短くするためには ℓ を出来るだけ小さくすべきではあるが, 逆に ℓ を大きくした方が速く収束することもある. そのような場合には ℓ を大きくすることは有益となる.

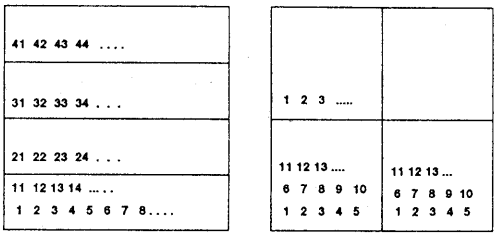
例えば BiCGStab(2) 法を BiCGStab2 法と比較すると, 全体の計算量, 記憶領域において BiCGStab(2) は BiCGStab2 に比べて計算量でおよそ 3 割, メモリの量で 1 割少なくて済む.

4. 並 列 化

BiCGStab 法関連の算法は行列・ベクトル・スカラ演算の組合せで構成されている. 今回これらのコードを並列化するにあたり, 算法そのものを並列化するのではなく, 行列とベクトルの積, 内積などの関数をそれぞれ並列化した. 以下では特に行列とベクトルの積の計算の並列化について考える.

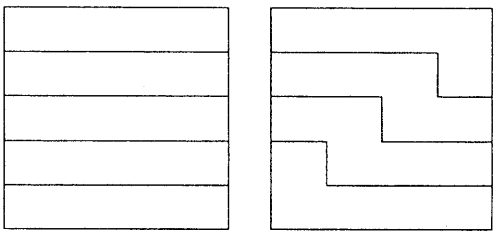
与えられた領域において 5 点中心差分法で離散化して得られる行列について考える. この行列の計算を並列化する時いくつかの方法が考えられる. 以下ではメッシュの大きさを MESH × MESH , セルの数を CELL とする, また, 1 つのセルが担当するベクトルの大きさを VEC = MESH × MESH/CELL とする. ここでは MESH×MESH が CELL で割り切れる場合を考えるがそれ以外の場合にも一般化が容易である.

(1) 与えられた領域を整合順序 (Natural Ordering) で番号づけして, それを先頭から順に各セルに割り当てる. これを示しているのが図 1(a)である. MESH が



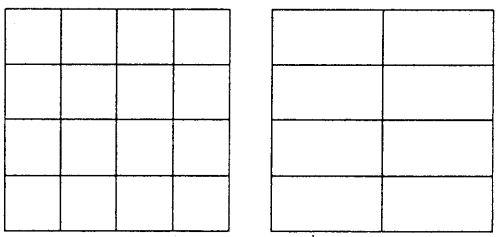
(a) 整合順序による番号付け　(b) ブロック分割による番号付け

図 1　並列化の方法



(a) MESH mod CELL = 0　(b) MESH mod CELL ≠ 0

図 2　整合順序による領域の分け方



(a) CELLX = CELLY　(b) CELLX ≠ CELLY

図 3　ブロック分割による領域の分け方

CELL で割り切れる場合には領域は図 2(a) のように, それ以外の場合には図 2(b)のように重複の無い領域に端から区切られる. 計算を行う場合それぞれのセルは隣のセルとの境界上の要素 を必要とする. 領域を MESH×MESH で区切った場合, 隣のセルと MESH 分だけの通信を必要とする. 前後のセルとの通信を考えると 1 セルあたり 2×MESH 分だけの通信が必要となる. ベクトルの大きさに対する通信の割合

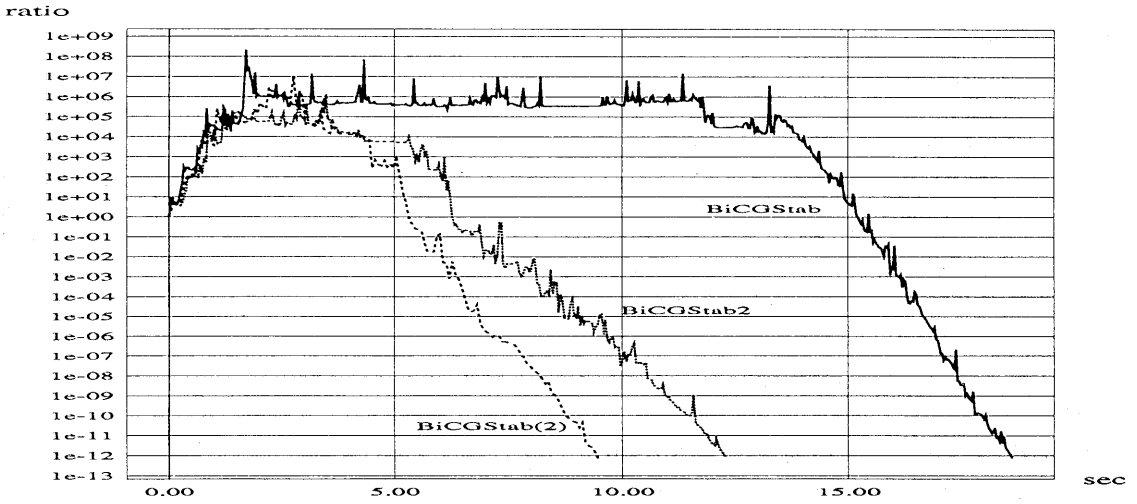


図 4　いろいろな BiCGStab における残差の収束性

は 2 × (CELL/MESH) となる.

(2)　使用するセルを 2 次元に配置して x 方向のセルの数を CELLX , y 方向のセルの数を CELLY とする. ここでは MESH は CELLX , CELLY の両方で割り切れるものとする.

与えられた領域を重複しない同じ形をした正方形 (図 **3**(a)) または長方形 (図 **3**(b)) に区切る. CELLX = CELLY の時, 区切られた領域は図 3(a)のように正方形になる. そしてこの領域を 2 次元で配列されたセルにそれぞれ割り当てる. その分けられた領域ごとに図 1(b)のように整合順序で番号付けをして計算を行う. ここではこれを**ブロック分割を用いた番号付け**と呼ぶことにする. この時セルが担当するベクトルの大きさは (MESH / CELLX) × (MESH / CELLY) である. これは (1) の場合と同じ大きさである.

一方, 計算時の通信量については隣のセルとの境界部分の要素を必要とするから x 方向のセルについては MESH/CELLY, y 方向については MESH/CELLX の通信量を必要とする. 特に CELLX = CELLY の時の 4 方向の通信量を考えると 4 × (MESH / CELLX) の通信量が必要となる. CELL = CELLX × CELLY を考えると, ベクトルの大きさに対する通信量の割合は 4 ×($\sqrt{\text{CELL}}$ / MESH) となる.

セルの数が増えていく場合, 整合順序によって番号付けすると O(CELL) で通信の割合が増えていくのに対し, ブロック分割によって番号付けすると通信の割合は $O(\sqrt{\text{CELL}})$ で増えていく.

つまりメッシュの大きさを固定してセルの数を増やしていく場合, ブロック分割した方が通信の割合が小さくて済む. これはセルの数を多くしていくにつれて違いが顕著になっていく.

5. 数値実験

矩形領域 $\Omega = [0,1] \times [0,1]$ における 2 階の楕円型偏微分方程式のディリクレ境界条件問題

$$u_{xx} + u_{yy} + a(x,y)u_x + b(x,y)u_y + 50u = f(x,y),$$
$$u(x,y)\mid_{\partial\Omega} = 0.$$

に対して解を $u(x,y) = (1-e^x)(1-x)y(1-e^{1-y})$ と設定し, 右辺を決定する. これを 5 点中心差分近似を用いて離散化し数値実験を行った.

各算法の初期近似解は零ベクトル, 収束条件は $\|r_k\|_2/\|r_0\|_2 \leq 1.0 \times 10^{-12}$ とした. また計算には, 富士通の並列計算機 AP1000 (最大 64 プロセッサ) を使用した.

プロセッサの数, メッシュの大きさ, $a(x,y)$, $b(x,y)$ の値, 用いる方法についてはそれぞれの実験ごとに決めるものとする.

5.1　いろいろな BiCGStab 法の収束性の比較

次のような条件での各算法における残差の収束の様

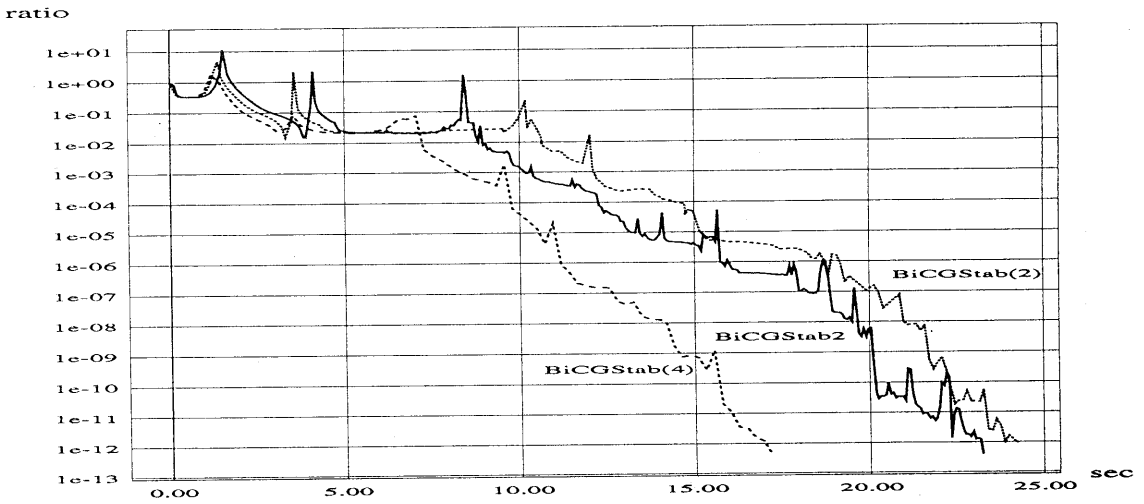


図 5　BiCGStab(ℓ) どうしの比較

子を図 4 に示す.

- $a = 4.0,\ b = 500.0$
- メッシュ 128×128
- プロセッサ数　64
- 解法：　BiCGStab, BiCGStab2, BiCGStab(2)
- 並列化の方法　　ブロック分割

BiCGStab2 法の一回の反復で必要な内積計算は BiCGStab(2) 法の 2 倍以上で，これは計算の並列化には不利となる．　無限の精度で計算した場合，BiCGStab2 法は BiCGStab(2) 法と同じ残差ベクトル列を生じる．一方，BiCGStab2 法の 1 回の反復に要する計算時間は BiCGStab(2) 法の約 1.2 倍 (実測値) である．図 4 の残差ベクトルの収束性を見るとこれがそのまま結果に反映されていることがわかる．

5.2　BiCGStab(ℓ) どうしの比較

次のような条件で ℓ の値を変えた時の BiCGStab(ℓ) どうしの残差の収束の様子を比較した．

- $a = 400xy(1-y)$, $b = 3xy(1-x)$
- メッシュ　128×128
- プロセッサ数　64
- 解法：BiCGStab2, BiCGStab(2), BiCGStab(4)
- 並列化の方法　　ブロック分割

BiCGStab(ℓ) 法は ℓ の値が大きくなると，ベクトルの直行化に要する計算量が $O(\ell^2)$ で増えていく．多くの場合 ℓ の値をいくつにしても残差の収束に要する反復の回数は同じぐらいになる．そのような場合には ℓ を小さくした方が良い．しかし ℓ を大きくした方が速く収束する場合もある．

図 5 は BiCGStab(4) 法が BiCGStab(2) 法や BiCGStab2 法に比べて速く収束している例である．

5.3　並列化の方法の比較

次のような条件で並列化の方法を変えてそれぞれの残差の収束時間を比較した．

- $a = 3.0,\ b = 50.0$
- メッシュ　64×64, 128×128
- プロセッサ数　1 〜 64
- 解法：　BiCGSTAB(2)

メッシュ 128×128 , セルの数が 64 の場合の並列化の方法を変えたときの収束時間の違いを図 6 に示す．それぞれの並列化の方法に対し単純に一定反復当りにかかる時間を比べると，整合順序による番号付けをした場合はブロック分割による番号付けをした場合の 1.1 倍である．これはメッシュ 128×128 , セル 64 個の場合の実測値である．しかし実際にはそれぞれのセルが受け持つベクトルの要素が異なるために，内積をとる際に丸め誤差が生じ，収束時間はこの通りにはならない．

次にメッシュ 64×64 , 128×128 に対してセルの数を変化させてその台数効果 (並列化効果) を比較する．

図 7 はメッシュ 128×128 の場合の並列化効果 (台数効果) である．この場合それぞれの並列化効果の違いは

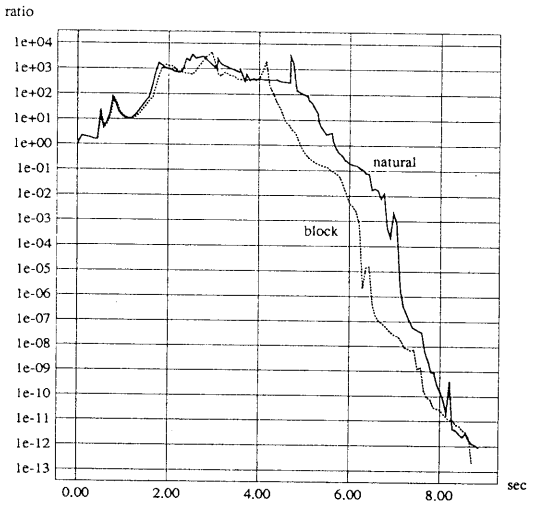


図 6　セルが 64 個の時のオーダリングの比較
メッシュ128 × 128

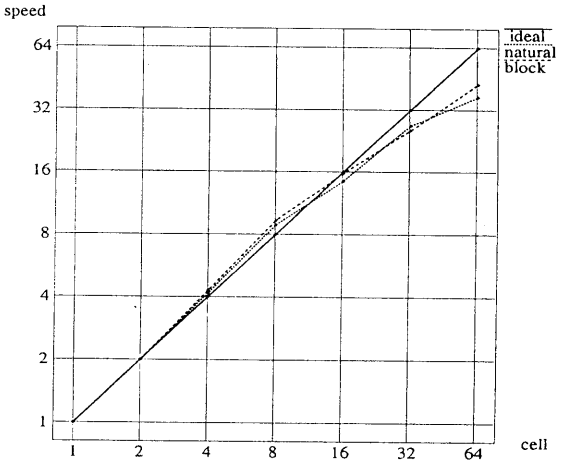


図 7　メッシュ 128 × 128 の場合の台数効果

それほど大きくない．それに対しメッシュ 64×64 の場合の並列化効果を示した図 8は，メッシュ 128×128 の場合と比べて違いが大きくなっている．この理由は前述した通りである．

6. おわりに

BiCGStab 関連のアルゴリズムを並列化したが，ここで行った数値実験では BiCGStab(ℓ) 法が非対称な連立 1 次方程式に対して効果的な解法である．

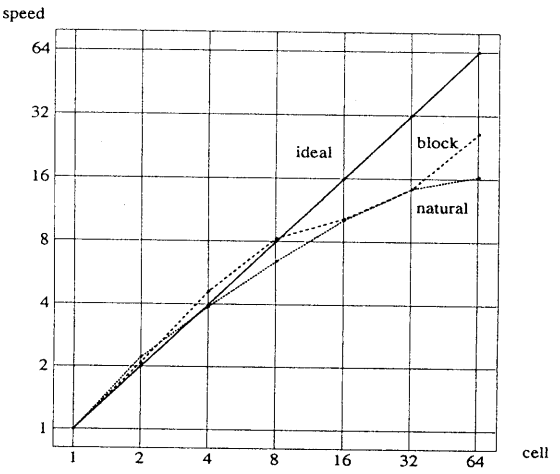


図 8　メッシュ 64 × 64 の場合の台数効果

BiCGStab(2) 法は正確に計算を行えば BiCGStab2 法と全く同じ反復を行う．しかしこれまでの議論や数値実験から BiCGStab(2) 法の方が優れていると言える．

セル間の通信の問題から線形な台数効果は得られなかったがメッシュMESH×MESH を細かくする時や，セルの数を大きくする場合には，通信の割合が減り並列化の効果は高くなる．並列化の方法としては領域をブロック分割した方が優れている．この違いはセルの数を増やしていくにつれて大きくなる．

参 考 文 献

1) H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13(1992), pp. 631-644.

2) M. H. GUTKNECHT, *Variants of BiCGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput., 14(1993), pp. 1020-1033.

3) G. L. G. SLEIJPEN and D. R. FOKKEMA, *BiCG-STAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, ETNA 1(1993), pp. 11-32.

4) 稲津、野口、野寺 「BiCGSTAB 系のいろいろなアルゴリズム」 NEC HPC 研究会 1995, pp. 21-23.

5) T. NODERA, A note on BiCGSTAB algorithm, SAUM., 18(1992), pp. 157-166.