ハイパフォーマンス
コンピューティング 77－20
（1999. 8. 3）

大規模疎行列係数連立一次方程式に対する前処理つき共役勾配法の並列化

塙　与志夫†　小柳義夫†

連立一次方程式に対する反復解法である Jacobi 法は，共役勾配法の前処理としても利用できることが知られている．この方法は，並列性が高く，元の問題の特定の性質に依存しない．本稿では，同様の性質を持つ前処理として，3 重対角部の Cholesky 分解に注目し，Jacobi 法を変形した方法を提案する．さらに，この前処理に適したスケーリング方法についても述べる．共有メモリ型並列計算機 Sun Ultra Enterprise 10000 上で評価した結果，多くの場合に Jacobi 法よりも提案した方法の方が有効であることがわかった．

Parallelization of the Preconditioned Conjugate Gradient Method for Large Scale Sparse Linear Systems

YOSHIO HANAWA† and YOSHIO OYANAGI†

The Jacobi method, which is an iterative method for solving linear systems, can also be used as a preconditioner for the conjugate gradient method. It has enough scalability, and does not require any properties of the original problem and uses only the matrix form. In this paper, we propose another preconditioner which is a variant of the Jacobi method related with the Cholesky decomposition of the tridiagonal part of a matrix. We also present a scaling method specialized for our preconditioner. Our experiment on the shared memory parallel computer Sun Ultra Enterprise 10000 shows that our preconditioner performs better than the Jacobi preconditioner in many cases.

1. はじめに

移流項のない偏微分方程式を離散化して得られる，疎で対称な大規模連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

に対しては，前処理つき共役勾配法（Preconditioned CG 法，PCG 法[1]）が有効な解法の一つとして知られている．PCG 法の速度は，利用する前処理による収束率の改善と，前処理部分の計算時間に依存する．逐次計算においては，PCG 法の中でも，不完全 Cholesky 分解を前処理に用いる方法（ICCG 法[2]）が高速である．しかし，並列計算においては，共役勾配法自体が高い並列性を持っているため，前処理にも収束の速さのみならず高い並列性が求められる．従って，通常のオーダリングでは並列性が低い ICCG 法は，単純に実装したのでは並列計算機には適さない．一方で，元の問題にある空間的局所性を利用することにより，並列性の高さと収束の速さと同時に実現した前処理も提案されているが，ライブラリに組み込む場合などのように，離散化やオーダリングの性質がわからない場合には適用できない．そこで，本研究では，行列形式のみを利用する並列性の高い前処理として，3 重対角部の Cholesky 分解に基づく Jacobi 法を変形した定常的な反復法を提案・実装した．また，数値実験により，点 Jacobi 前処理と比較して収束が速いことを示した．

† 東京大学 大学院 理学系研究科 情報科学専攻
Department of Information Science, Graduate School of Science, University of Tokyo

2. 前処理つき共役勾配法と並列化

2.1 前処理つき共役勾配法

PCG 法の基礎反復法である共役勾配法（CG 法）は，誤差の A ノルムを小さくしていくことで連立一次方程式の近似解を求める方法で，行列が正定値対称のときに適用可能である．この近似解は，初期解 $\boldsymbol{x}^{(0)}$ から探索方向ベクトル $\boldsymbol{p}^{(k)}$ に沿っての最小化を繰り返すことで求められる．CG 法の反復式は，初期解 $\boldsymbol{x}^{(0)}$，$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$，$\boldsymbol{p}^{(0)} = \boldsymbol{r}^{(0)}$ としたとき次のように書ける．

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)} \tag{2}$$

$$\boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - \alpha_k A\boldsymbol{p}^{(k)} \tag{3}$$

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)} \tag{4}$$

$$\alpha_k = \frac{(r^{(k)}, r^{(k)})}{(p^{(k)}, Ap^{(k)})} \tag{5}$$

$$\beta_k = \frac{(r^{(k+1)}, r^{(k+1)})}{(r^{(k)}, r^{(k)})} \tag{6}$$

行列の変形を必要とせず，行列ベクトル積が計算できればよいので，疎な連立一次方程式に適しているが，行列の性質によっては収束が悪く，利用しにくいことも多い．

これに対し，PCG 法は対称な前処理行列 $K = LL^T$ を解くべき方程式 (1) に作用させた，

$$L^{-1}AL^{-T}(L^T x) = L^{-1}b \tag{7}$$

という方程式に対して CG 法で解くことと等価な方法である．この式を整理し，反復式を CG 法同様に書けば，初期解 $x^{(0)}$, $r^{(0)} = b - Ax^{(0)}$, $p^{(0)} = K^{-1}r^{(0)}$ として

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)} \tag{8}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)} \tag{9}$$

$$p^{(k+1)} = K^{-1}r^{(k+1)} + \beta_k p^{(k)} \tag{10}$$

$$\alpha_k = \frac{(r^{(k)}, K^{-1}r^{(k)})}{(p^{(k)}, Ap^{(k)})} \tag{11}$$

$$\beta_k = \frac{(r^{(k+1)}, K^{-1}r^{(k+1)})}{(r^{(k)}, K^{-1}r^{(k)})} \tag{12}$$

となる．これを式 (2) ～ (6) と比較すると，1 反復あたり $K^{-1}r$ の計算が 1 回増えているだけであり，この計算の増加による影響よりも収束が改善すれば，全体としての計算時間は短くなる．一般に K は A に近いほど収束は速くなるので，K に求められる性質は，A に近いことと $K^{-1}r$ が容易に求まることである．K として A の不完全 Cholesky 分解を用いる ICCG 法は，$K^{-1}r$ の計算が行列ベクトル積 Ap と同程度と，計算量が少ない割に収束が改善するので，疎で対称正定値である連立一方程式の解法として広く用いられている．

2.2 並列化と前処理

式 (8) ～ (12) より，PCG 法に表れる計算はベクトルの内積，ベクトル同士の和，行列ベクトル積，前処理行列の逆行列とベクトルの積，の 4 種類からなる．最初の 3 つは容易に並列化できるが，逆行列とベクトルの積は必ずしも並列に実行できない．例えば，ICCG 法では前進後退代入によって逆行列とベクトルとの積を計算するが，行列の構造が既知でない場合，これを並列に実行することは難しい．多くの前処理方法では，1 反復あたりの計算量に対する前処理の割合は無視できないので，前処理部分の並列性が低い方法より，多少収束が悪くても並列性の高い前処理の方が，全体として高い性能を出すことが多い．今回提案する方法も，ICCG 法に比べると収束は悪いものの，並列性が高い方法である．

2.3 前処理としての定常的な反復法

ここでは，定常的な反復法を CG 法の前処理とすることを考える．初期値 $x^{(0)}$ とし，行列 A を $A = M - N$ と分割すると，

$$Mx^{(k+1)} = Nx^{(k)} + b \tag{13}$$

という形の反復法が得られる．このような反復法について，初期値 $x^{(0)} = 0$ としたときに得られる反復解を $x^{(k)} = X^{(k)}b$ と表わすとすると，

$$X^{(1)} = M^{-1}, \quad X^{(k+1)} = M^{-1}(NX^{(k)} + I)$$

であるから，式 (13) で M, N が対称であれば，この反復法の k 回反復は CG 法の前処理として利用できる．M を A の対角成分 D とした反復法である点 Jacobi 法をはじめ，SSOR 法などの定常的な反復法を何回か反復した近似が CG 法の前処理に利用できることは以前から知られているが，並列計算に適用する場合には，その並列性から点 Jacobi 法が主に使われる．

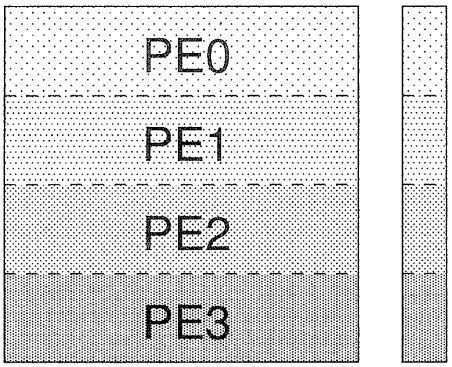

図 1 データ分割

3. 提案する前処理と実装

3.1 データ分割

並列化を考えるとき，各 Processor Element (PE) に対するデータの分割を考える必要がある．今回は全ての行列とベクトルについて，図 1 のように一次元ブロック分割を行なった．行列ベクトル積が主な計算であること，また，提案する前処理を考えると，このような分割が最も自然であると考えられる．

3.2 切り欠きつき 3 重対角ヤコビ法

今回提案する前処理は，A の 3 重対角部のうち，対称な位置の要素を同じ PE が担当しているような部分 T の完全 Cholesky 分解に基づく，定常的な反復法を利用する．データ分割を一次元ブロック分割としたので，この T の形状は図 2 のように，A の要素のうちで各 PE の受け持つ行でブロック分割したときの，対角ブロック内の 3 重対角部となる．提案する方法は，式 (13) において $M = T$ とした定常的な反復法を $Ax = r$ について 2 回反復した式

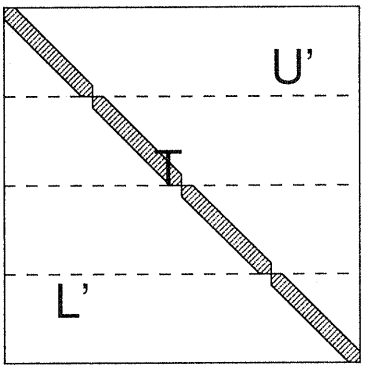

図2 提案する方法での行列の分割

$$\boldsymbol{x}^{(2)} = T^{-1}(T - L' - U')T^{-1}\boldsymbol{r} \qquad (14)$$

を $K^{-1}\boldsymbol{r}$ として用いるものである．

この方法は，線の一部が途中で切れているような線 Jacobi 法とみなすこともできる．以下ではこの方法を切り欠きつき3重対角ヤコビ法 (Dented Tridiagonal Jacobi method, DTJ) と呼ぶ．この DTJ 法は線方向について一度の反復で厳密解を求めることができるため，全ての方向の緩和に1反復前の値を使う点 Jacobi 前処理に比べると収束が良いと期待できる．

また，元の問題を仮定したときの線 Jacobi 法と，DTJ 法とを比べると，線上で別の PE の持つ点との緩和には1反復前の値を使うことになるが，このことにより，3重対角行列 T の Cholesky 分解の前進後退代入において PE 間の通信がないため，並列性も点 Jacobi 前処理と遜色ない．通信を避けるための T の切り欠きが多くなればなるほど点 Jacobi 法に近付くので，PE の数が増えるほど収束は悪くなるはずだが，実験的には多くの問題で PE の数に拘らず似た収束を示す．直感的に，線 Jacobi 法において緩和の一部を点 Jacobi 法と同じように計算しても，その結果は線 Jacobi 法と点 Jacobi 法の中間程度の性能だろうということである．しかし，その予想に反し，前処理として使ったときに点 Jacobi 法より悪くなるような問題もある．

この前処理の1反復あたりの計算量をより単純な方法である点 Jacobi 前処理と比較する．点 Jacobi 法の2回反復の式は A の上三角行列，下三角行列，対角行列をそれぞれ U, L, D とすると

$$\boldsymbol{x}^{(2)} = D^{-1}(D - L - U)D^{-1}\boldsymbol{r} \qquad (15)$$

と書ける．これと式 (14) との大きな違いは T^{-1} が D^{-1} になっていることであるが，D^{-1} の計算は対角行列の積であり，浮動小数点乗算1回であるのに対し，T^{-1} の計算は2重対角行列の前進代入・後退代入により，浮動小数点乗算4回，加算2回と多い．また，DTJ 法では T^{-1} を Cholesky 分解した結果を保持しておく必要があるので，記憶領域の点でも不利になっている．

3.3 スケーリング

連立一次方程式のスケーリングとは，解くべき方程式 (1) に対し，適当な対角行列 S を作用させ，

$$(S^{-1/2}AS^{-1/2})S^{1/2}\boldsymbol{x} = S^{-1/2}\boldsymbol{b} \qquad (16)$$

とし，$\tilde{A} = (S^{-1/2}AS^{-1/2})$, $\tilde{\boldsymbol{x}} = S^{1/2}\boldsymbol{x}$, $\tilde{\boldsymbol{b}} = S^{-1/2}\boldsymbol{b}$ として，スケーリングされた方程式 $\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ を解くような方法である．式 (7) から PCG 法を導いた際とは異なり，行列やベクトル自体を変更して新たな方程式を考えるので，スケーリング後の方程式に対してさらに PCG 法を適用することもできる．多くの場合，$S = D$ とする対角スケーリングにより，$(S^{-1/2}AS^{-1/2})$ の対角要素を全て1にすることで，行列ベクトル積での浮動小数乗算の回数とメモリアクセス回数を減少させることができる．ここで，対角スケーリングした方程式に点 Jacobi 前処理を使った CG 法を行なうことを考えると，対角スケーリング後の対角成分は1であるので，式 (15) のうち D と D^{-1} が I となり，演算回数，メモリアクセス回数の点で有利となる．これに対し，DTJ 前処理では T^{-1} の対角成分が1であっても Cholesky 分解すると1以外の数になってしまうので，点 Jacobi 前処理ほど有利にはならない．これは問題であるので，以下で，DTJ 前処理に向いたスケーリングを考えるが，まず，DTJ 前処理がスケーリングについて不変であることを示す．

スケーリングをした連立一次方程式 (16) に対してさらに前処理行列 $\tilde{K}$ で前処理を行なったとき，式 (7) より

$$\begin{aligned}&(\tilde{K}^{-T/2}[S^{-1/2}AS^{-1/2}]\tilde{K}^{-1/2})(\tilde{K}^{1/2}[S^{1/2}\boldsymbol{x}])\\ &\quad = \tilde{K}^{-T/2}[S^{-1/2}\boldsymbol{b}]\end{aligned}$$

となる．これは，$A\boldsymbol{x} = \boldsymbol{b}$ に対し，前処理行列 $K = S^{1/2}\tilde{K}S^{1/2}$ を用いることと等価である．

スケーリング後の式 $\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ の $\tilde{A}$ の成分を図2での T, L', U' それぞれにに対応する成分 $\tilde{T}, \tilde{L}', \tilde{U}'$ に分割する．このとき，

$$\begin{aligned}\tilde{T} &= S^{-1/2}TS^{-1/2}\\ \tilde{L}' &= S^{-1/2}L'S^{-1/2}\\ \tilde{U}' &= S^{-1/2}U'S^{-1/2}\end{aligned}$$

となる．スケーリング後の式に対する前処理行列 $\tilde{K}$ は

$$\begin{aligned}\tilde{K} &= \tilde{T}(\tilde{T} - \tilde{L}' - \tilde{U}')^{-1}\tilde{T}\\ &= S^{-1/2}T(T - L' - U')^{-1}TS^{-1/2}\end{aligned}$$

となり，スケーリング前の式に対して同じ前処理を用いたときと等価である．この議論は点 Jacobi 前処理についても同様にできる．以上の議論より，DTJ 前処理ではどんなスケーリングを行なっても反復ごとの収束は変化しないので，対角スケーリングよりも計算を省略できるスケーリングがあれば，そのスケーリングを利用した

方が有利である．

T を Cholesky 分解した 2 重対角行列の対角部が全て 1 となるようなものは，そのような望ましいスケーリングとなっていることを示す．スケーリング行列を S，スケーリング後の 3 重対角部を $\tilde{T} = S^{-1/2} T S^{-1/2}$ とする．S の (i,i) 成分を s_i とし，$\tilde{T}$ の Cholesky 分解 $\tilde{L}\tilde{L}^T$ について，$\tilde{L}$ の (i,i) 成分を 1，$(i-1,i)$ 成分を c_i となるとする．$\tilde{T}$ と T を以上の変数を用いて表すと，

$$\tilde{T} = \begin{pmatrix} 1 & c_2 & & \mathbf{0} \\ c_2 & \left(1+c_2^2\right) & \ddots & \\ & \ddots & \ddots & c_n \\ \mathbf{0} & & c_n & \left(1+c_n^2\right) \end{pmatrix} \quad (17)$$

$$T = \begin{pmatrix} s_1^2 & c_2 s_1 s_2 & & \mathbf{0} \\ c_2 s_1 s_2 & \left(1+c_2^2\right) s_2^2 & \ddots & \\ & \ddots & \ddots & c_n s_{n-1} s_n \\ \mathbf{0} & & c_n s_{n-1} s_n & \left(1+c_n^2\right) s_n^2 \end{pmatrix}$$

となる．T の非零要素数が $(2n-1)$ 個であるのに対し，変数も $s_1, \cdots, s_n, c_2, \cdots, c_n$ の $(2n-1)$ 個であるので，このようなスケーリングは可能である．これらの変数は実際に以下のようにして求めることができる．まず T の $(1,1)$ 成分から s_1 を求める．s_k が求まっているならば T の $(k+1,k)$ 成分から $c_{k+1}s_{k+1}$ が求まり，T の $(k+1,k+1)$ 成分と $c_{k+1}s_{k+1}$ から s_{k+1} が求まるので，c_{k+1} も求まる．行列 A が優対角であることから，これらの変数はすべて実数となる．以上より，帰納的に全ての変数が求められる．

このスケーリングにより，T^{-1} の計算での前進代入・後退代入のそれぞれで 1 行について浮動小数点乗算 1 回が省略でき，乗算 2 回，加算 2 回で計算できる．スケーリングをした点 Jacobi 前処理 CG 法では $D^{-1} = I$ となり，この部分の計算はないので，1 反復 1 行あたりでいうと乗算 4 回，加算 4 回の差があるものの，スケーリングによる計算回数の減少についてみると点 Jacobi 法のスケーリングより効果は大きい．

また，T について記憶する必要のある値は A の $(i-1,i)$ 成分 c_i だけとなり，記憶領域についてもスケーリングにより利益を得られる．なぜなら，T の Cholesky 分解は c_i と 1 からなるので，T^{-1} の計算は c_i だけで可能である．また，式 (17) より，T の行列ベクトル積の計算についても c_i だけで計算できることがわかる．

これを単純に実装しただけでは，行列の対角成分を各反復ごとに計算する必要があるが，行列ベクトル積での計算順序を変更することで，計算量を変えずに実装できる．以下，式 (17) の c_i と $\boldsymbol{p}$ の i 番目の要素を p_i を用いて説明する．$\tilde{T}\boldsymbol{p}$ の i行目の計算で $c_i p_{i-1} + (1+c_i^2) p_i$ を計算するが，これを $c_i(p_{i-1} + c_i p_i) + p_i$ のように計算する．$(i-1)$ 行目の計算で A の $(i-1,i)$ 要素 c_i と p_i の積を計算するので，これを保存しておけば，$c_i(p_{i-1} + c_i p_i) + p_i$ の計算は浮動小数点乗算 1 回加算 2 回で可能である．仮に，$1 + c_i^2$ を d_i として計算しておくとすると，この部分は $c_i p_{i-1} + d_i p_i$ となり，浮動小数点乗算 2 回加算 1 回と d_i の記憶領域へのアクセスが必要なので，むしろ不利になる．

以上の工夫により，スケーリングをした点 Jacobi 前処理 CG 法（以下，PJCG）とスケーリングをした DTJ 前処理 CG 法（以下，DTJCG）とで，必要な記憶領域は同程度となる．両者で異なる部分は 3 重対角部に関するデータだけであるが，この部分について PJCG で持つ必要があるのは A の (i,i) 成分だけであるので，3 重対角部が十分密であれば必要な記憶領域は同じである．

3.4 データ構造

このような反復法において，必要なデータはスカラーとベクトルと行列のいずれかであり，最初の 2 つはそれぞれ浮動小数点数とその配列として実現できる．行列については，その性質により適当なデータ構造は変わるが，今回は疎行列を対象としているので，それぞれの行について非零要素の列の index と値の組を配列とすることにした．ただし，DTJCG では 3 重対角部の Cholesky 分解をするので，3 重対角部は零要素を含めて別の配列に取り，行列の残りの要素のみを各行について列と値の組で持たざるを得ない．今回の数値実験問題では 3 重対角部が比較的密であること，また前処理自体の性能の比較が目的であることから，他の前処理も同様にデータを持つようにした．

4. 数値実験

数値実験として，並列ソフトウェア・コンテスト PSC'98[3] の問題から，特徴的な問題を用いて反復回数による残差の現象の様子を PJCG などの従来の方法と DTJCG について比較した．

問題 1（PSC98 予選問題 4）は，拡散係数が急激に変化するような問題で，2 次元単位領域におけるディリクレ境界条件を持つポアソン方程式

$$-\nabla \cdot (k \nabla u) = f \quad \text{in } \Omega = (0,1) \times (0,1)$$
$$\text{with } u = 0 \text{ on } \partial\Omega$$

に対し，領域を 512×512 のメッシュで離散化を行った問題である．拡散係数 k は $(0.25, 0.25) \times (0.75, 0.75)$ の領域で $k = 100$，残りの部分で $k = 1$ である．問題 2（PSC98 予選問題 3）は，3 次元の巨大な問題で，$(0,1) \times (0,2) \times (0,2)$ の，拡散係数が一定の 3 次元領域におけるポアソン方程式を，$64 \times 128 \times 128$ のメッシュで 7 点差分を用い離散化を行った問題である．問題 3（PSC98 予選問題 5）は，直接法が速いような問題で，ディリクレ境界条件を持つ 1 次元のポアソン方程式を，16384 のメッシュで離散化を行った問題である．それぞれの問題の格子点のオーダリングは辞書式順序で

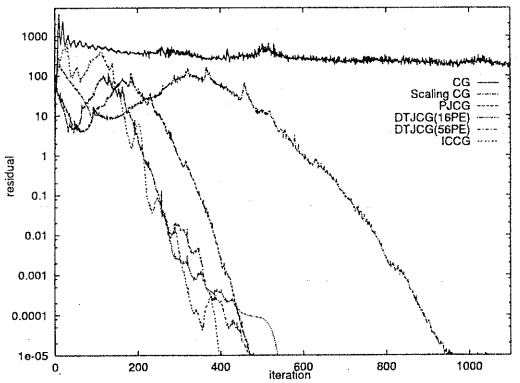

図 3 問題 1 での反復回数と残差の減少

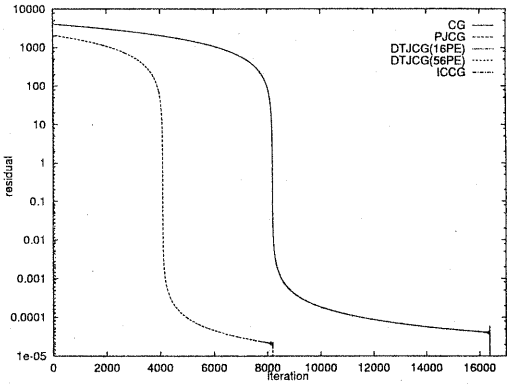

図 5 問題 3 での反復回数と残差の減少

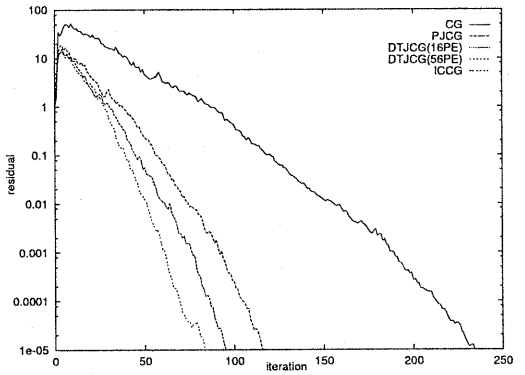

図 4 問題 2 での反復回数と残差の減少

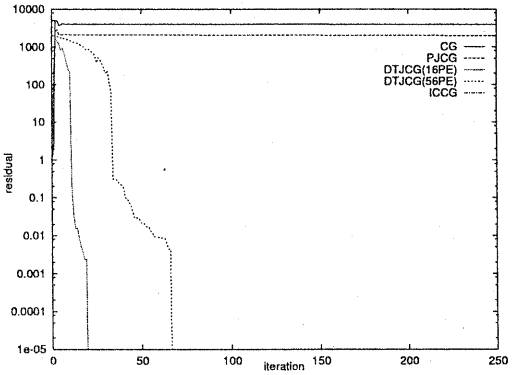

図 6 図 5 の 250 回反復まで

あり，右辺は格子番号 i に対し $0.5\sin(i+1)$ である．

まず，反復回数と残差の関係について実験した．それぞれの問題について，前処理なしの CG 法と，問題 1 のみは scaling CG （対角部スケーリングを行なった前処理なしの CG 法），PJCG 法，今回提案した DTJCG 方について 16PE と 56PE のそれぞれと，並列には実行できないが，比較のため ICCG 法と，5 ～ 6 種類を実験した．その様子を示したのが図 3 ～ 図 6 である．ここでの残差は PSC98 の問題の終了条件と同じく無限大ノルムであるが，一般に，数反復から数十反復後より後は，2- ノルムと無限大ノルムのどちらのノルムでも相対的に同様な挙動を示す．

問題 1 （図 3）は反復回数の少ない順に ICCG, DTJCG(62PE), PJCG, DTJCG(16PE), Scaling CG の順である．図では切れているが，前処理なしの CG 法は約 7000 反復後に残差が 1e-5 になっている．この図で特徴的なことは，16PE の DTJCG が 2-step PJCG よりも収束が悪くなっていることである．DTJCG は点 Jacobi 法と線 Jacobi 法の中間的な方法であるが，両方の方法より悪くなることもあるといえる．また，PE 数が多い方が，より収束が悪くなる傾向が予想されるが，逆転することがあることもわかる．一方で，62 PE での DTJCG は PJCG よりも収束が速く，300 反復程度までは ICCG と拮抗していることから，DTJCG 法はこのような解きにくい問題に関して有効である可能性があり，さらにその性質を調べる必要があるといえる．

問題 2 （図 4）は ICCG, DTJCG, PJCG, CG の順に収束が速い．DTJCG は 16PE, 62PE の両グラフがほぼ重なっている．この問題では DTJCG は ICCG よりは悪いものの，PJCG より収束が速い．

問題 3 （図 5）は他の解法に比べて PJCG と CG が極端に遅い．0 付近を拡大したのが（図 6）である．この図でも ICCG のグラフは見えないが，問題 3 では不完全 Cholesky 分解が完全分解と等しいので，前処理で方程式が解けてしまっているためである．

これらの 3 問を含む，PSC'98 の予選と本選で使われた 10 問について，CG 法，2-step PJCG 法，DTJCG 法の 3 つの方法で，残差が 1e-5 になるまでの実行時間を比較した．これらは，最初の 2 つについては対角部スケーリングを，DTJCG 法については提案したスケーリングをそれぞれ利用した．計算機としては，

表 1　各解法 (56PE) による実行時間 (秒)

	本選 1	本選 2	本選 3	本選 4	本選 5	予選 1	予選 2	予選 3	予選 4	予選 5
scaling CG	140.420	11.220	175.198	3.287	88.141	5.468	5.946	9.934	6.248	75.939
PJCG	115.091	8.500	161.387	3.642	49.874	3.675	3.425	8.835	4.304	30.242
DTJCG	97.291	7.767	163.718	3.490	0.790	3.532	3.263	8.379	5.097	0.764

表 2　各問題の特徴

	本選 1	本選 2	本選 3	本選 4	本選 5	予選 1	予選 2	予選 3	予選 4	予選 5
次元	2 次元	2 次元	2 次元	3 次元	1 次元	2 次元	2 次元	3 次元	2 次元	1 次元
差分化	5 点差分	5 点差分	9 点差分	7 点差分	3 点差分	5 点差分	5 点差分	7 点差分	5 点差分	3 点差分
拡散係数	一様	非一様	一様	非一様	非一様	一様	一様	一様	非一様	一様

Sun Ultra Enterprise10000 (UltraSPARC 250MHz ×64) を用い，うち 56 PE を使用した．その結果が表 1 である．各問題の性質は先の 3 問でほぼ代表されていると思われるが，参考のため，問題の性質を簡単にまとめたのが表 2 である．拡散係数の欄が非一様となっているものは，どれも一部の領域で拡散係数が急激に変化するような問題である．

この結果から，多くの問題で DTJCG 法の方が実行時間が短くなっていることがわかる．これと 図 3 ～ 図 6 を見比べると，収束の改善に比べて実行時間の改善の方が小さいことから，1 反復あたりの計算時間は増加していることがわかる．これは，同期やループの回数など，前処理全体の複雑さが原因であろう．逆に，スケーリングのみの CG 法は，1 反復あたりの時間が小さいので，必要な反復回数が多いにも関わらず，問題によっては他の方法に匹敵することがわかる．

本選問題 5 は予選問題 5 と同様に 1 次元 Poisson 方程式から導かれる 3 重対角行列の問題であるが，どちらも図 5 から予想されるように，DTJCG 以外では非常に時間がかかっている．ライブラリなどで使う場合には，帯行列に対しては別の解法を使うなどできるので，これだけでは DTJCG が優れているとは結論できないが，3 重対角行列に限らず，同様の性質を持った問題があるとすれば有効だといえよう．また，この 2 問を除いた 8 問の計算時間を合計してもなお DTJCG が最も速いことから，総合的にみて DTJCG は有効な方法だといえる．

5. おわりに

定常的な反復法を CG 法の前処理に使う方法は以前から知られていたが，式 (13) の M を複雑にするほど計算が複雑になるため，点 Jacobi 法以外の定常的な反復法が，multi-coloring などのオーダリングをせずに並列計算機での共役勾配法の前処理として使われることは少なかった．この研究では，Jacobi 法を変形した並列性の高い前処理を提案し，スケーリングにより，計算量を大きくせずに反復を改善できることを示した．

今後の課題としては，まず，同様に並列性の高い方法[4] との比較を行うことが挙げられる．

また，DTJ 前処理では，点 Jacobi 法に比べて収束が悪くなる問題があるが，図 7 のように，わずかに減速すると性質が良くなることが実験的にわかっている．このような性質を説明するため，固有値分布など，この前処理の性質をより詳細に調べることが必要である．

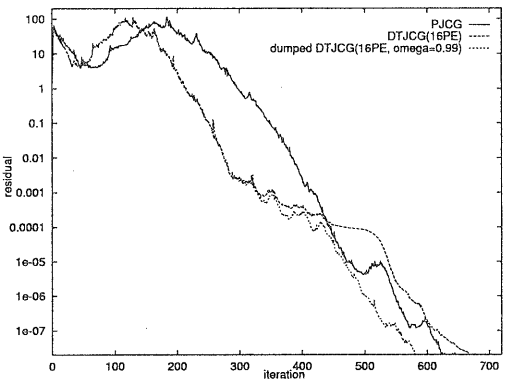

図 7　問題 1 での減速した DTJ 法の残差の減少

また，DTJCG 法は 3 重対角部が密であるほど有効であるので，3 重対角部が疎な問題のために，3 重対角部を密にするリオーダリングの実装も課題である．

参考文献

1) P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations,* in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, des., Academic Press, NewYork, 1976, pp. 309-332.

2) J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix,* Math. Comp., 31(1977), pp. 148-162.

3) http://www.kudpc.kyoto-u.ac.jp/HPC-WG/PSC98/

4) L. Yu. Kolotilina and A. Yu. Yeremin. *Factorized Sparse Approximate Inverse Preconditionings I. Theory* SIAM. J. Matrix Anal. Appl., 14(1993), pp. 45-58.