解　説

大規模行列計算における反復解法†

名取　亮††　野寺　隆†††

1. はじめに

スーパコンピュータの出現によって，流体計算，VLSI設計，計算物理などにおいて大規模な計算が行われるようになった．このような分野では，自然現象をモデル化して偏微分方程式を構成し，それを有限要素法や有限差分法で離散近似することによって最終的には連立1次方程式を解くことに帰着されることが多い．この連立1次方程式の係数は一般に大型疎行列である．

連立1次方程式の解法には，行列要素を消去してゆくガウスの消去法のタイプに属する直接法と，適当な第0近似解から出発して反復公式を何度も繰り返し適用して次第に精度を良くしてゆく SOR 法などの反復解法がある．反復解法の利点は，疎行列のパターンをいつまでも保つため記憶容量が増加しないことである．しかも，主な計算は行列とベクトルの積の計算であり，ベクトル化も容易である．

共役勾配法は反復解法の利点をもちながら，高々n回（nは行列の次数）のステップで厳密解に収束するという直接法的な性質も兼ね備えた効率のよい解法である．特に，行列の前処理と併用すれば，その収束性を格段に高めることができるため，近年注目を集めている．これについては **2.** で詳しく論ずる．

さらに，今までほとんど計算不可能とさえいわれていた大規模な非対称行列系の問題に対しても，ベクトルプロセッサを効率よく利用できる反復解法がいろいろ考案されている．それらについては **3.** で述べることにする．**4.** では，これらの反復解法をスーパコンピュータで行うときに注意すべき点について述べる．

† Iterative Methods for Large Scale Matrix Probrems by Makoto NATORI (Institute of Information Sciences and Electronics, University of Tsukuba) and Takashi NODERA (Department of Mathematics, Keio University).

†† 筑波大学電子情報工学系
††† 慶應義塾大学理工学部数理科学科

2. 対称正定値行列に対する反復解法

共役勾配法は1952年に Hestenes & Stiefel[1] によって開発された算法で，対称正定値行列Aを係数とする連立1次方程式

$$Ax=b \qquad (1)$$

に適用できる．この方法の原理は(1)の解が2次関数

$$F(x)=(x, Ax)-2(x, b) \qquad (2)$$

を最小にするものであることに基づいている．ここで，(x, y)はベクトル x と y の内積を表す．また，残差

$$r=b-Ax$$

を用いて誤差関数

$$E(x)=(r, A^{-1}r) \qquad (3)$$

を定義すると，

$$E(x)=F(x)+(b, A^{-1}b)$$

で，右辺第2項は定数であるから，$F(x)$ を最小にすることと $E(x)$ を最小にすることは同じである．共役勾配法の算法は次のようになる．

[共役勾配法の算法]

(1) 任意の初期値 x_0 を選び，残差ベクトル $r_0=b-Ax_0$ を計算し，$p_0=r_0$ とおく．

(2) 収束するまで次の手順を繰り返す．($k=0, 1, 2, \ldots$)

$$a_k=(r_k, r_k)/(p_k, Ap_k)$$
$$x_{k+1}=x_k+a_kp_k$$
$$r_{k+1}=r_k-a_kAp_k$$
$$b_k=(r_{k+1}, r_{k+1})/(r_k, r_k)$$
$$p_{k+1}=r_{k+1}+b_kp_k$$

収束判定は，相対残差 $\|r_k\|/\|b\|$ がある値 ε より小さくなったことで行う．

このようにして計算された残差ベクトル r_k は，Krylov 列 $\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\}$ によって張られる部分空間の中のベクトルで誤差関数を最小にするものになっている．そして，直交条件

$$(r_i, r_k)=0, \quad (i\neq k) \qquad (4)$$

を満たしている．各反復での探索方向を示すベクトル p_k は，A に関する共役直交条件

$$(p_i, Ap_k)=0, \quad (i\neq k) \qquad (5)$$

を満足している．n次元空間における線形独立なベクトルが高々n個であることから，$r_n=0$ となることがわかる．これは共役勾配法が高々n回で厳密解に収束することを示している．

共役勾配法の収束の様子を調べてみると，問題によってはn回の反復ステップよりもずっと少ない反復回数で良い近似解に収束する場合がある．共役勾配法の収束性は解を構成するための固有ベクトルに対応した固有値分布に強く依存する．係数行列の固有値が縮重していたり，密集していたりすると，収束が速くなるのである．したがって，行列の固有値分布を改善するような行列の前処理 (preconditioning) を施してやれば，共役勾配法の収束性を格段に向上させることが可能である．

行列の前処理：$n\times n$ の正則行列 P, Q を用いて，方程式(1)に同値な次の方程式を考える．

$$P^{-1}AQ^{-1}(Qx)=P^{-1}b \qquad (6)$$

ここで，(1)を解く代わりに

$$B=P^{-1}A\ Q^{-1},\ c=P^{-1}b$$

とおいて

$$By=c$$

を解き，その後で

$$Qx=y$$

から近似解 x を計算する．重要な問題は，行列 P, Q の選択である．当然のことながら行列Aのなんらかの情報をもとに決定可能で，行列 B の条件数が行列 A の条件数より良くなるように決める必要がある．しかも，P, Q ともに逆行列が簡単に計算できるものでなければ実用にならない．そのために，対角行列や三角行列がよく用いられる．Aが対称正定値のときには，Bもまた対称正定値にして共役勾配法が使えるようにするのがよい．そのためには，P は正定値とし，$Q=P^T$ とすればよい．

行列の前処理は，対角行列スケーリング，行列分離 (matrix splitting) によるもの，および行列分解 (matrix decomposition) に基づくものに大別できる．

対角行列スケーリング：Aの対角要素からなる対角行列Dを

$$D=\mathrm{diag}\,[a_{11}, a_{22},\ldots, a_{nn}]$$

とする．Aは正定値だから対角要素はすべて正である．そこで

$$D^{-1/2}=\mathrm{diag}\,[a_{11}^{-1/2}, a_{22}^{-1/2}, \ldots, a_{nn}^{-1/2}]$$

を用いて，(1)の方程式を

$$D^{-1/2}AD^{-1/2}(D^{1/2}x)=D^{-1/2}b$$

と変形する．$D^{-1/2}AD^{-1/2}$ の対角要素はすべて 1 である．これが最も手軽な前処理である (Bauer[2])．

行列分離による前処理：対角行列スケーリングを行った方程式を改めて $Ax=b$ と書く．係数行列は

$$A=L+I+U$$

と分離できる．ただし，Lは狭義下三角行列，Iは単位行列，Uは狭義上三角行列である．もし，Aが対称ならば $U=L^T$ である．ここで，方程式を

$$(I+\omega L)^{-1}A(I+\omega U)^{-1}[(I+\omega U)x]=(I+\omega L)^{-1}b$$

のように変換する．これは

$$P=I+\omega L$$
$$Q=I+\omega U$$

と選んだ前処理である．Aが対称ならば $U=L^T$ であるから $Q=P^T$ となっている．ω は調整パラメータといわれるもので，連立 1 次方程式の反復解法である SOR 法の加速パラメータと類似の変数である．Evans[3] によれば，ω は $0\leqq\omega<2$ の範囲を動くことができ，この範囲で

$$B=(I+\omega L)^{-1}A(I+\omega U)^{-1}$$

の条件数を最小にするものが取れる．そして，Bの条件数をAの条件数の平方根程度まで減少させることができる．最適な ω を決定することはなかなか難しいが，算法が簡単でありその効果も大きいので，なかなか捨てがたい方法である．この前処理と共役勾配法の大型行列計算への応用が Axelsson[4] と野寺[5] にある．

行列分解による前処理：A が対称正定値の場合には，コレスキー分解によって下三角行列Lと上三角行列 L^T の積

$$A=LL^T$$

に分解することができる．コレスキー分解の算法は次のようになる．

[コレスキー分解の算法]

$k=1$ から n に対して以下の演算を行う

$$l_{kk}=\left(a_{kk}-\sum_{j=1}^{k-1}l_{kj}^2\right)^{1/2}$$

$i=k+1$ からnに対して以下の演算を行う

$$l_{ik}=\left(a_{ik}-\sum_{j=1}^{k-1}l_{ij}l_{kj}\right)\Big/l_{kk}$$

ここで重要な問題は，行列Aの非ゼロ要素のパターンである．行列Aが疎行列であっても，分解行列Lは一般に密行列になる．行列が小さければそれほど問題は

ないが，大型である場合には計算量および記憶容量が増大して，実行困難となる．そこで，分解行列Lも行列Aと同程度に疎となるような不完全コレスキー分解を考える必要が生まれてくる．

不完全コレスキー分解は Meijerink & van der Vorst[6] によって考えられた．彼らは2階の自己随伴楕円型偏微分方程式を離散化した方程式に対して，不完全コレスキー分解と共役勾配法を併用することによって安定でかつ収束性の高い算法を構成した．現在この算法は ICCG (Incomplete Cholesky Conjugate Gradient) 法とよばれ，物理，工学の諸分野で広く用いられている．

不完全コレスキー分解では，行列Lの要素のなかで強制的にゼロにする場所のインデックス集合

$$P=\{(i,\ j)|l_{ij}=0\}$$

をあらかじめ設定しておき，コレスキー分解の計算で$(i,\ j)\in P$ となる l_{ij} は何も計算せずにゼロと置く．行列Aの要素がゼロである場所のインデックス集合を

$$P_A=\{(i,\ j)|a_{ij}=0\}$$

としたとき，$P=P_A$ と選ぶのが最も単純でかつ有効な方法である．さらに，行列Lの非ゼロ要素を増やすことも考えられる[7]．

行列 A が M 行列 ($i\neq j$ のとき $a_{ij}\leqq 0$, かつ $A^{-1}>0$) ならば，どんな P を用いても不完全コレスキー分解が可能である[6]．したがって，P を一度決めれば，一つの不完全コレスキー分解

$$A=LL^T+R$$

が決定できることになる．ただし，Rは行列Lの中で要素を強制的にゼロにしたことによって生ずる誤差を表す．不完全コレスキー分解の算法は次のようになる．

[不完全コレスキー分解の算法]

$k=1$ から n に対して以下の演算を行う

$$l_{kk}=\left(a_{kk}-\sum_{j=1}^{k-1}l_{kj}^2\right)^{1/2}$$

$i=k+1$ から n に対して

もし $(i,\ k)\in P$ ならば $l_{ik}=0$ とし，そうでなければ以下の演算を行う

$$l_{ik}=\left(a_{ik}-\sum_{j=1}^{k-1}l_{ij}l_{kj}\right)\Big/l_{kk}$$

この不完全行列分解を用いて，方程式(1)を

$$L^{-1}AL^{-T}(L^Tx)=L^{-1}b$$

のように変換する．これは

$$P=L,\ Q=L^T$$

と選んだ前処理である．LL^T が A の近似行列になっていれば，$L^{-1}A\,L^{-T}$ は近似的に単位行列になるから共役勾配法の収束は速い．ICCG 法の算法は次のようになる．

[ICCG 法の算法]

(1) 任意の初期値 x_0 を選び，残差ベクトル $r_0=b-Ax_0$ と $P_0=(LL^T)^{-1}r_0$ を計算する．

(2) 収束するまで以下の手順を繰り返す．($k=0,\ 1,2,\ldots$)

$$a_k=((LL^T)^{-1}r_k,\ r_k)/(p_k,\ Ap_k)$$
$$x_{k+1}=x_k+a_kp_k$$
$$r_{k+1}=r_k-a_kAp_k$$
$$b_k=((LL^T)^{-1}r_{k+1},\ r_{k+1})/((LL^T)^{-1}r_k,\ r_k)$$
$$p_{k+1}=(LL^T)^{-1}r_{k+1}+b_kp_k$$

ここで，$z=(LL^T)^{-1}r$ の計算は二つの方程式

$$Ly=r \qquad (7)$$
$$L^Tz=y \qquad (8)$$

を解くことによって行う．スーパコンピュータ（ベクトル計算機）で実行するときには，この計算を並列化するための工夫が必要となる．これについては **4.** で述べる．

以上が ICCG 法の原理であるが，実際には演算量の節約のために

$$A=LDL^T+R$$

の形の不完全コレスキー分解が用いられる．Dは対角行列で，普通の改訂コレスキー分解とは異なりDの対角要素 d_i は $d_i=1/l_{ii}$ とする．また，この不完全コレスキー分解を補正した MICCG 法 (Modified ICCG method) が Gustafsson[8] により提案され，ICCG 法よりも収束が速いことが確かめられている[9]．

3. 非対称行列のための反復解法

行列Aが非対称の場合には，共役勾配法は使えないので，その代わりに共役残差法 (Conjugate Residual Method, CR 法)，双共役勾配法 (Biconjugate Gradient Method, BCG 法) あるいは自乗共役勾配法 (CG Squared Method, CGS 法)，チェビシェフ反復法などが用いられる．

共役残差法：この方法は

$$G(x)=(b-Ax,\ b-Ax)=(r,\ r)$$

を最小にするxを逐次探索する方法である．xが(1)の解のとき $G(x)$ は最小値0をとる．共役残差法の算法は次のようになる．

[共役残差法の算法]

(1) 任意の初期値 x_0 を選び，残差ベクトル

$r_0=b-Ax_0$ を計算し $p_0=r_0$, $q_0=Ap_0$ とする.

(2) 収束するまで, 次の手順を繰り返す. ($k=0,1,2,\ldots$)

$$a_k=(r_k,\ Ap_k)/(Ap_k,\ Ap_k)$$
$$x_{k+1}=x_k+a_kp_k$$
$$r_{k+1}=r_k-a_kAp_k$$
$$b_k=-(Ar_{k+1},\ Ap_k)/(Ap_k,\ Ap_k)$$
$$p_{k+1}=r_{k+1}+b_kp_k$$
$$q_{k+1}=Ar_{k+1}+b_kq_k$$

このようにして計算されたベクトル r_k, p_k に対しては

$$(Ap_{k+1},\ Ap_k)=0$$
$$(r_{k+1},\ Ar_k)=0$$

が成り立つ. 算法を変更して

$$b_{ik}=-(Ar_{k+1},\ Ap_i)/(Ap_i,\ Ap_i),\quad i=0,1,\ldots,k$$
$$p_{k+1}=r_{k+1}+\sum_{i=0}^{k}b_{ik}p_i$$

とすれば

$$(Ap_i,\ Ap_k)=0,\quad (i\neq k)$$
$$(r_i,\ Ar_k)=0,\quad (i\neq k)$$

が成り立つので, n 回の反復で解が得られる. しかし, この方法では p_i をすべて保存しておくために記憶容量が増える上に計算量も多くなるので, 普通は初めに示した算法が使われている.

共役残差法は, Aの対称部

$$M=(A+A^T)/2$$

が正定値ならば収束し

$$\frac{(r_{k+1},\ r_{k+1})}{(r_k,\ r_k)}\leqq 1-\frac{\{\lambda_{\min}(M)\}^2}{\lambda_{\max}(A^TA)}$$

が成り立つことが知られている. この式からも, 前処理によって収束を速くできることがわかる.

双共役勾配法: この方法は(1)の方程式と, それに双対な方程式

$$A^Tx^*=b^*$$

を組み合わせて計算する方法である. (b^* と b は等しくとるのが普通である.) 双共役勾配法の算法は次のようになる.

[双共役勾配法の算法]

(1) 任意の初期値 x_0 と x_0^* を選び, 残差ベクトル $r_0=b-Ax_0$ および $r_0^*=b^*-A^Tx_0^*$ を計算し, $p_0=r_0$, $p_0^*=r_0^*$ とおく. (x_0 と x_0^* とは0とすることが多い.)

(2) 収束するまで次の手順を繰り返す. ($k=0,1,2,\ldots$)

$$a_k=(r_k^*,\ r_k)/(p_k^*,\ Ap_k)$$
$$x_{k+1}=x_k+a_kp_k$$
$$r_{k+1}=r_k-a_kAp_k;\quad r_{k+1}^*=r_k^*-a_kA^Tp_k^*$$
$$b_k=(r_{k+1}^*,\ r_{k+1})/(r_k^*,\ r_k)$$
$$p_{k+1}=r_{k+1}+b_kp_k;\quad p_{k+1}^*=r_{k+1}^*+b_kp_k^*$$

双共役勾配法は次のようにして導くことができる. $2n$ 次の行列とベクトル

$$\tilde{A}=\begin{bmatrix}A & 0\\ 0 & A^T\end{bmatrix},\quad \tilde{x}=\begin{bmatrix}x\\ x^*\end{bmatrix},\quad \tilde{b}=\begin{bmatrix}b\\ b^*\end{bmatrix}$$

を用いて, 方程式

$$\tilde{A}\tilde{x}=\tilde{b}\qquad (9)$$

を考え, 内積

$$\langle\tilde{x},\ \tilde{y}\rangle_{\tilde{H}}=(\tilde{x},\ \tilde{H}\tilde{y})=(\tilde{H}\tilde{x},\ \tilde{y})$$

を定義する. ただし

$$\tilde{H}=\begin{bmatrix}0 & I\\ I & 0\end{bmatrix}$$

である. $\tilde{A}$ はこの内積に対して自己随伴

$$\langle\tilde{x},\ \tilde{A}\tilde{y}\rangle_{\tilde{H}}=\langle\tilde{A}\tilde{x},\ \tilde{y}\rangle_{\tilde{H}}$$

だから, 関数

$$\tilde{F}(\tilde{x})=\langle\tilde{x},\ \tilde{A}\tilde{x}\rangle_{\tilde{H}}-2\langle\tilde{x},\ \tilde{b}\rangle_{\tilde{H}}$$

を停留にする $\tilde{x}$ が(9)の解になることがわかる. この停留値を共役勾配法で計算すると双共役勾配法の算法が得られる. 詳しくは名取[10]を見よ.

$\tilde{r}_k$ と $\tilde{p}_k$ の生成多項式をそれぞれ $R_k(\tilde{A})$, $P_k(\tilde{A})$ とすると

$$\tilde{r}_k=R_k(\tilde{A})\tilde{r}_0$$

から

$$r_k=R_k(A)r_0;\quad r_k^*=R_k(A^T)r_0$$

が, また

$$\tilde{p}_k=P_k(\tilde{A})\tilde{r}_0$$

から

$$p_k=P_k(A)r_0;\quad p_k^*=P_k(A^T)r_0^*$$

が得られる. そして, 直交性

$$\langle\tilde{r}_i,\ \tilde{r}_k\rangle_{\tilde{H}}=0\quad (i\neq k)$$
$$\langle\tilde{p}_i,\ \tilde{A}\tilde{p}_k\rangle_{\tilde{H}}=0\quad (i\neq k)$$

から, 共役直交性

$$(r_i^*,\ r_k)=0\quad (i\neq k)$$
$$(p_i^*,\ Ap_k)=0\quad (i\neq k)$$

が導かれる. これによって, 双共役勾配法も高々 n 回の反復で解に収束することが分かる. ただし, 正定値性がないため, 途中で a_k の分母がゼロになって計算が続けられなくなる危険性がある. そういう場合には, 初期値を変えてやりなおせばよい.

自乗共役勾配法：これは双共役勾配法の変種で，den Heijer[11] がデバイスシミュレーションの計算に用いて良好な結果を得た．この方法は，双共役勾配法の算法を変形して，新しいベクトル列 $\hat{r}_k$ と $\hat{p}_k$ を計算するようにしたものである．ここで

$$\hat{r}_k = R_k^2(A)\hat{r}_0$$

$$\hat{p}_k = P_k^2(A)\hat{r}_0$$

である．ただし，$R_k(A)$ と $P_k(A)$ は双共役勾配法における r_k と p_k の生成多項式である．自乗共役勾配法の算法は次のようになる．

[自乗共役勾配法の算法]

（1） 任意の初期値 x_0 を選び，残差ベクトル $\hat{r}_0 = b - Ax_0$ を計算し，$\hat{p}_0 = e_0 = \hat{r}_0$ とおく．

（2） 収束するまで次の手順を繰り返す．($k=0,1,2,\ldots$)

$$a_k = (\hat{r}_0,\ \hat{r}_k)/(\hat{r}_0,\ A\hat{p}_k)$$

$$h_{k+1} = e_k - a_k A\hat{p}_k$$

$$\hat{r}_{k+1} = \hat{r}_k - a_k A(e_k + h_{k+1})$$

$$x_{k+1} = x_k + a_k(e_k + h_{k+1})$$

$$b_k = (\hat{r}_0,\ \hat{r}_{k+1})/(\hat{r}_0,\ \hat{r}_k)$$

$$e_{k+1} = \hat{r}_{k+1} + b_k h_{k+1}$$

$$\hat{p}_{k+1} = e_{k+1} + b_k(h_{k+1} + b_k\hat{p}_k)$$

ここで，e_k と h_{k+1} は $\hat{r}_k$ と $\hat{p}_k$ を計算するために必要な補助ベクトルで

$$e_k = p_k(A)R_k(A)\hat{r}_0$$

$$h_{k+1} = P_k(A)R_{k+1}(A)\hat{r}_0$$

である．この算法と双共役勾配法との対応についての説明は名取[10]にある．

この算法では，$\|\hat{r}_k\|$ が小さくなれば，x_k が良い近似解になる．もし，双共役勾配法において

$$\|r_k\| = \|R_k(A)r_0\|$$

が小さくなったとすると，自乗共役勾配法における

$$\|\hat{r}_k\| = \|R_k^2(A)r_0\|$$

はもっと小さくなることが期待できるので，自乗共役勾配法は双共役勾配法よりも速く収束する．

適応的チェビシェフ法：非対称行列系の反復解法の一つに，Manteuffel[12),13)] により提案された適応的なチェビシェフ法がある．チェビシェフ多項式は，数値解析の多くの分野で利用されているが，大型の線形問題の解法にも利用することができる．すなわち，チェビシェフ多項式には，次のような効果的な性質があるからである．

（1） 反復パラメータを適切に選択すれば，残差多項式のノルムは急速に減少し，チェビシェフ法の収束は速い．

（2） チェビシェフ多項式の三項漸化式から，近似解 $\{x_k\}$ の系列を計算するための計算コストのかからない反復式を導出できる．

チェビシェフ法の基本公式は，

$$x_{k+1} = -\alpha_k A x_k + (1+\beta_k)x_k - \beta_k x_{k-1} + \alpha_k b$$

で与えられる．また，係数 α_k，β_k は，次のように定義すればよい．

$$\alpha_k = \frac{2T_k(d/c)}{cT_{k+1}(d/c)},\quad \beta_k = \frac{T_{k-1}(d/c)}{T_{k+1}(d/c)}$$

ただし，$T_k(z) = \cosh(k\cosh^{-1}(z))$ である．パラメータ c と d は，チェビシェフ法の収束をつかさどるものであり，行列 A のスペクトラムに従属するものである．よって，もし適切な反復パラメータ c, d の値が与えられるならばチェビシェフ反復は次のように行えばよい．

[チェビシェフ法の算法]

（1） 初期値 x_0 を選び，残差ベクトル

$$r_0 = b - Ax_0$$

および

$$p_0 = (1/d)r_0$$

を計算する．

（2） 次の反復を繰り返す．($k=0,1,2,\ldots$)

$$x_{k+1} = x_k + p_k$$

$$r_{k+1} = b - Ax_{k+1}$$

$$\alpha_{k+1} = \begin{cases} 2d/(2d^2-c^2), & k=0 \\ 1/(d-(c/2)^2\alpha_k), & k\geqq 1 \end{cases}$$

$$\beta_{k+1} = d\alpha_{k+1} - 1$$

$$p_{k+1} = \alpha_{k+1}r_{k+1} + \beta_{k+1}p_k$$

チェビシェフ法も，他の解法と同様に，多項式に基づく反復解法であるので，k 回反復における残差ベクトルは，生成多項式を用いて次のように書き表せる．

$$r_k = R_k(A)r_0$$

ただし，

$$R_k(z) = \frac{T_k((d-z)/c)}{T_k(d/c)}$$

通常，$R_k(z)$ は残差多項式と呼ばれている．Manteuffel[2] は，反復が進むにしたがい，この残差多項式に次の関係があることを示した．

$$R_k(z) \fallingdotseq S(z)^k$$

ただし，

$$S(z) = \frac{(d-z)+[(d-z)^2-c^2]^{1/2}}{d+(d^2-c^2)^{1/2}}$$

この場合，反復パラメータ c と d は，中心を d とし

て，$d\pm c$ を焦点とする楕円の族を決定することになる．すなわち，この楕円族に対して，行列Aのスペクトラムを含む最小な楕円が存在し，この楕円がチェビシェフ法の漸近収束率 $|S(z)|$ を決定するのである．よって，チェビシェフ法が最適な収束をするためには，次のミニ・マックス問題を解くことにより，反復パラメータを決定すればよいことになる．

$$\min_{d,c}\ \max_{z\in sp(A)}\ |S(z)|$$

ただし，$sp(A)$ は行列のスペクトラムを示すものとする．しかし，当然，行列Aのスペクトラムは未知なのだから，この値を推定しなければならない．Manteuffel[13] は，係数行列Aの対称部が正定値という条件のもとで，チェビシェフ反復から得られた情報から，これらの値をダイナミックに推定する算法を提案した．すなわち，適当なパラメータを利用して，可能な初期値からチェビシェフ反復をスタートし，残差の収束性をモニタする．もし収束が思わしくないときには，反復により得られた情報をもとにして固有値を推定し，新しい反復パラメータを計算する．この値を用いてチェビシェフ反復をリスタートする．この適応的な過程は，よい反復パラメータが得られるまで繰り返す．その後は，チェビシェフ反復のみを行い解を求めることになる．算法は次のようになる．

[適応的チェビシェフ法]

（1） チェビシェフ法の算法を繰り返し，その収束性をモニタする．もし，収束が思わしくない場合（残差ベクトルが発散したり，収束が遅いとき），反復を停止し，新しい反復パラメータ c, d を選択する次の操作を行う．

（2） 適応的な過程

(2.1) 行列Aのスペクトラムの凸包で固有値を推定する．

(2.2) これらの固有値を含む最適な楕円を求め，新しい反復パラメータを決定する．

（3） 新しい反復パラメータが古いパラメータと同じ場合には，そのままチェビシェフ反復を続行する．そうでない場合には，新しい反復パラメータを用いて，チェビシェフ反復をリスタートする．

Manteuffel[13] は，固有値の推定に修正べき乗法を用いているのだが，これを別の操作で置き換えても問題はない．たとえば，Arnordi 法などを利用して固有値を求めてもよいのである．このような算法をハイブリッド法と呼んでいる[14]．

行列の前処理：これらの方法を用いる場合にも，**2.** で述べたような前処理を施して収束を速めることができる．対角行列スケーリングと行列分離による前処理については **2.** で述べたとおりである．行列分解に基づく前処理に対しては，Aが非対称行列の場合には不完全コレスキー分解の代わりに不完全 LU 分解 (Incomplete LU Decomposition, ILU 分解) を用いればよい．

不完全 LU 分解では，分解行列LとUの要素の中で強制的にゼロにする場所のインデックス集合Pをあらかじめ設定しておいて，LU 分解の計算で $(i, j)\in P$ となる l_{ij} および u_{ij} は何も計算せずにゼロとおく．この場合にも，$P=P_A$ と選ぶのが最も単純かつ有効である．不完全 LU 分解の算法は次のようになる．

[不完全 LU 分解の算法]

$k=1$ から n に対して以下の演算を行う

$l_{kk}=1$

$j=1$ から $k-1$ に対して

もし，$(k, j)\in P$ ならば $l_{kj}=0$ とし，

そうでなければ以下の演算を行う

$$l_{kj}=\left(a_{kj}-\sum_{i=1}^{j-1} l_{ki}u_{ij}\right)\Big/u_{jj}$$

$$u_{kk}=a_{kk}-\sum_{i=1}^{k-1} l_{ki}u_{ij}$$

$j=k+1$ から n に対して

もし，$(k, j)\in P$ ならば $u_{kj}=0$ とし，

そうでなければ以下の演算を行う

$$u_{kj}=a_{kj}-\sum_{i=1}^{k-1} l_{ki}u_{ij}$$

これは，L の対角要素を 1 とする標準の LU 分解であるが，ICCG 法のときと同じように，対角行列Dを用いて LDU 分解を行い，D の対角要素 d_k として $d_k=1/u_{kk}=1/l_{kk}$ となるものを採用すると演算量を節約することができる．

不完全分解による前処理を行う場合には

$$Ax=b$$

を解く代わりに

$$(LU)^{-1}Ax=(LU)^{-1}b$$

を解くことになる．係数行列 $(LU)^{-1}A$ が単位行列に近くなるので，反復回数が大幅に減少する．

対称行列のときと同様に，方程式を

$$L^{-1}AU^{-1}(Ux)=L^{-1}b$$

と変形することもできる．行列 $(LU)^{-1}A$ と $L^{-1}AU^{-1}$ は相似であるから，固有値分布は全く同じである．し

たがって，この章で述べた反復解法のどれに対しても収束特性は本質的に同じである．ただし，残差ベクトルが $(LU)^{-1}r$ となるか $L^{-1}r$ となるかの違いがある．したがって，見掛け上収束の様子が異なってみえる．実は，前者の方が優れている点が二つある．第一点は $(LU)^{-1}$ が A^{-1} の近似であることから $(LU)^{-1}r$ は $A^{-1}r$，すなわち誤差に近いので，誤差を直接0にできる点である．第二点は，プログラム上後者にくらべて作業ベクトルが1本少なくてすむ点である．

4. スーパコンピュータと反復解法

共役勾配法は元来ベクトル計算機向きの算法なのであるが，不完全分解による前処理を行うときにはベクトル化のための工夫が要る．ここでは，直方体領域における一般化ポアソン方程式

$$\mathrm{div}\,(-k\ \mathrm{grad}\ u)=f$$

を7点差分公式で離散化した方程式を例として説明する．格子点番号は辞書順，すなわち x, y, z 方向の順につけることにすると，係数行列 A は，図-1 のようになる．ただし添字は要素の列番号に対応してつけるものとする．$P=P_A$，すなわち非零要素の形を変えない不完全コレスキー分解 (LDT^T 分解) の下三角行列 L は図-2 のようになる．ここで，a_i, b_i, c_i は A と同じであることに注意しよう．対角行列 D の要素 d_i は次の式によって計算される．

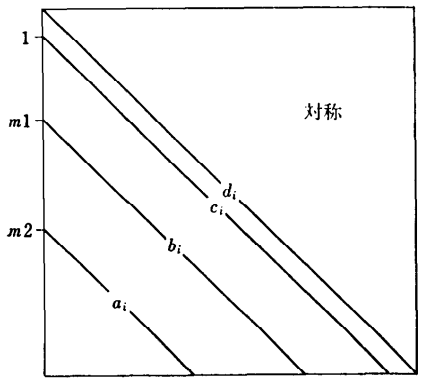

図-1 行列 A

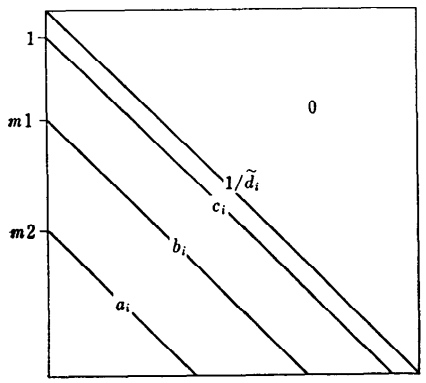

図-2 行列 L

$$d_i=1/(d_i-c_{i-1}{}^2d_{i-1}-b_{i-m1}{}^2d_{i-m1}-a_{i-m2}{}^2d_{i-m2})$$

Gustafsson 流の補正をした不完全コレスキー分解では

$$\begin{aligned}d_i=1/\{&d_i-c_{i-1}{}^2d_{i-1}-b_{i-m1}{}^2d_{i-1}-a_{i-m2}{}^2d_{i-m^2}\\&-\alpha(a_{i-1}c_{i-1}d_{i-1}+b_{i-1}c_{i-1}d_{i-1}\\&\quad a_{i-m1}b_{i-m1}d_{i-m1}+c_{i-m1}b_{i-m1}d_{i-m1}\\&\quad b_{i-m2}a_{i-m2}d_{i-m2}+c_{i-m2}a_{i-m2}d_{i-m2})\}\end{aligned}$$

となる．ここで，補正係数 α は 0.975 ぐらいがよいとされている[15]．これらの前処理を施した CG 法を ICCG 法，MICCG 法という．アルゴリズムは **2.** で示したとおりである．このアルゴリズムでは単なる CG 法とは異なり，各反復ごとに $z=(LDL^T)^{-1}r$，すなわち二つの方程式

$$Ly=r \qquad (10)$$

$$DL^Tz=y \qquad (11)$$

を解かなければならない。(10)は前進代入，(11)は後退代入によって解くことができるが，直前の計算結果が右辺に現れるため並列性がない．不完全コレスキー分解の計算でも同様である．ただし，前進代入と後退代入は反復のたびに行わなければならないのに対して，不完全コレスキー分解は初めに1回行うだけである．これらに対しては，以下に述べるような超平面法によって並列化が可能である．

もとの問題に立ち帰って考えると，一つの格子点はその前後，左右，上下の隣接する格子点とだけ結合しているので，それに着目すると超平面法によってベクトル化できる[15),16)]．すなわち，格子点の x, y, z 座標をそれぞれ n_1, n_2, n_3 としたとき，$n_1+n_2+n_3=$一定の点は互いに独立であり，並列に計算できるのである．辞書順の格子点番号の代わりに $n_1+n_2+n_3=$一定であるような面（超平面）の番号と，超平面上の格子点番号を用いる．これらの対応づけはリストベクトルを用いて実現できる．詳しくは文献 15), 16) を参照されたい．複雑な形の領域を有限要素法で離散化したときの不規則なスパース行列に対しても，同じような考え方によってベクトル化することができる[15),16)]．

5. おわりに

本稿では，自然現象をモデル化した偏微分方程式を離散化して得られる大型疎行列の反復解法について述べた．問題は対称正定値行列と非対称行列に分類できる．対称正定値行列に対しては前処理付き共役勾配法が優れている．一方，非対称行列に対しては，いくつかの方法が提案されているが，現在のところ前処理付き自乗共役勾配法が良いようである．適応的チェビシェフ法についても今後さらに検討を加えるべきであろう．

参考文献

1) Hestenes, M. R. and Stiefel, E.: Method of Conjugate Gradient for Solving Linear Systems, J. Res. Nat. Bur. Stand., Vol. 49, pp. 409-436 (1952).
2) Bauer, F. L.: Optimally Scaled Matrices, Num. Math., Vol. 5, pp. 73-87 (1963).
3) Evans, D. J.: The Use of Preconditioning in Iterative Methods for Solving Linear Equation with Symmetric Positive Definite Matrices, J. IMA, Vol. 4, pp. 295-314 (1968).
4) Axelsson, O.: Solution of Linear Systems of Equations, Lecture Notes in Mathematics, No. 572, Springer-Verlag, pp. 1-51 (1977).
5) 野寺：大型行列に対する PCG 法，Seminar on Mathematical Science (Keio University), No. 7, マテマティカ (1983).
6) Meijerink, J. A. and van der Vorst, H. A.: An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-Matrix, Math. Comp., Vol. 31, pp. 148-162 (1977).
7) Meijerink, J. A. and van der Vorst, H. A.: Guide Lines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems, J. Comp. Phys., Vol. 44, pp. 134-155 (1981).
8) Gustafsson, I.: A Class of First Order Factorization Methods, BIT, Vol. 18, pp. 142-156 (1978).
9) 村田，小国，唐木：スーパコンピュータ，丸善，pp. 147-149 (1985).
10) 名取：BCG 法と CGS 法，京大数理研講究録，No. 613, pp. 135-143 (1987).
11) den Heijer, C.: Preconditioned Iterative Methods for Nonsymmetric Linear Systems, Proc. International Conference on Simulation of Semiconductor Devices and Processes, pp. 267-285 (1984).
12) Manteuffel, T. A.: The Tchebychev Iteration for Nonsymmetric Linear Systems, Num. Math., Vol. 28, pp. 307-327 (1977).
13) Manteuffel, T. A.: Adaptive Procedure for Estimating Parameter for Nonsymmetric Tchebychev Iterations, Num. Math., Vol. 31, pp. 390-398 (1978).
14) Elman, H. C., Saad, Y. and Saylor, P. E.: A Hybrid Chebyshev Krylov Subspace Algorithm for Solving Nonsymmetric Systems of Linear Equations, SIAM J. Sci. Stat. Comput., Vol. 7, pp. 840-855 (1986).
15) 後：ベクトル計算機向き ICCG 法，京大数理研講究録，No. 514, pp. 110-134 (1984).
16) 後，西方，長堀：スーパコンピュータ HITAC S-810 による行列計算，日立評論，Vol. 65, No. 8, pp. 557-562 (1983).

(昭和 62 年 6 月 17 日受付)