講　座

スパース行列処理技法 (1)*

大附辰夫** 川北建次**

1. はじめに

行列において，その非0要素の数が全体の要素数(行の数×列の数)に比べて十分少いとき，その行列は**スパース** (sparse)，あるいは**疎**であるという．線形連立方程式 $Ax=b$ の係数行列 A がスパースである場合に，この性質を利用して解析の能率を高めるための手法を総称して**スパース行列処理技法**という．通常のプログラミング手法によれば，A(次数を n とする.) は2次元配列の形式で記憶され，数値演算アルゴリズムは3重 DO ループを用いて記述される．この場合 n^2 に比例するメモリと n^3 に比例する計算時間を必要とし，n が数百以上になると，すぐに大型計算機の能力の限界を越えてしまう．スパース行列処理技法は A の非0要素に関する情報だけをコンパクトな形で記憶するためのデータ表現法と，無駄な計算 ($a+0=a$, $a\times0=0$ など)を省くための演算処理方式を基本とし，通常の方法では解けないような大規模（数百元～数万元）な連立方程式を対象とするものである．

スパース行列処理技法は線形方程式だけを対象とするのではなく，むしろ非線形方程式や微分方程式が介入するシステムの解析，設計において，より大きな威力を発揮するものである．というのは，非線形方程式求解のためのニュートン法や微分方程式求解のための数値積分法を計算機アルゴリズムとしてながめれば，線形方程式求解の繰り返しに過ぎないからである．このような意味で，スパース行列処理技法は大規模システムの解析，設計において必要不可欠な計算機利用技術であり，その応用範囲は電気回路，電力系統，機械振動系，建築構造物，流体輸送網などの工学の分野ばかりでなく，計量経済，エコロジーなどの社会科学の分野まで広がっている．また，応用数学の分野で独立した体系を成している偏微分方程式や数理計画法においても，スパース行列処理技法が重要な役割を演じている．

上記の応用分野のいずれにおいても，対象とする問題の規模が大きくなるほど行列のスパース性が増大するという傾向を持っている．このような応用に介入する線形方程式の係数行列においては，その次数に関係なく，一行当りの非0要素の数は2～10個程度であるといわれている．更に，多くの場合，A のスパース構造が一定のもとで数値データだけを変化させて $Ax=b$ を何回も解く必要があり，A のスパース構造をあらかじめ非数値処理によって分析しておいて，何回も繰り返される数値計算の手間を短縮させることを狙った技法が要求される．

行列のスパース性に着目した数値解法の歴史は，1950年代に，線形計画法[1,2]や偏微分方程式の解析[3~9]に適用されたことに端を発している．その後，1960年代後半から，線形方程式求解の基本であるところのガウス消去法（後述）において，行列のスパース性を利用する試みがなされ，その有効性が確められた．従来，大規模なスパース行列を係数とする連立方程式の求解は間接解法の受持ちとされていた***が，間接解法では収束する問題の範囲が限られており．また収束しても計算速度が遅く，丸め誤差に弱いなどの欠点がある．これを克服するための手法としてガウス消去法を前提としたスパース行列処理技法が注目されるようになった．特に，1968 年，IBM ワトソン・リサーチ・センタ（米国，ニューヨーク州）においてスパース行列処理技法だけを対象とした学会[10]が初めて開かれ，各界に大きな反響を及ぼしたのをきっかけとして，広範な応用分野をもつ共通基礎技術としての地位を築いた．この専門学会に刺激されてスパース行列処理技法を専攻する研究者が著しく増大し，この分野の第2回の専門学会[11]が 1970 年，オクスフォード大学(英国)

---

* Sparse Matrix Technology by Tatsuo OHTSUKI and Kenji KAWAKITA (Central Research Laboratories, Nippon Electric Co., Ltd.)

** 日本電気（株）中央研究所

*** 間接解法は係数行列にベクトルを掛けるという基本演算だけから成っており，係数行列に関する極く単純なデータ構造を用意することによって，その非0要素を能率よくアクセスすることができる．

において，第3回の専門学会[12]が 1971 年，IBM ワトソン・リサーチ・センタにおいて開催されるに至った．更に 1973 年には，スパース行列だけを取り扱った解説書[13]が初めて出版された．

本講座では，スパース行列処理技法において基本的に重要な事項を体系的にまとめて解説してみたい．しかし紙数の関係で全ての事項を網羅することはできず，また各々の事項の詳細に言及することもできないので，詳しくは末尾の参考文献を参照されたい．先ず第2章において，スパース行列処理の基本となる数値解析アルゴリズムを簡単に紹介する．次に，第3，4章において，行列のスパース性を分析する上で重要な非数値解析の問題（組み合わせ問題）およびそれに対するグラフ理論の応用について解説する．第 5〜7 章においては行列のスパース構造を取り扱うためのプログラミング技法について解説する．最後に，第8章において典型的な応用分野について言及する．

2. 行列解析の基礎

線形方程式（n 変数とする）

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \tag{2.1}$$

の解は，代数的には

$$\boldsymbol{x}=A^{-1}\boldsymbol{b} \tag{2.2}$$

で与えられる．しかし実際に逆行列 A^{-1} を求めて，これを $\boldsymbol{b}$ に掛けるという計算を行うのは最も拙劣な方法である．というのは A^{-1} を陽に求めるためには，後述のガウス消去法の3倍程度の計算時間を要し，それだけ丸め誤差の影響によって精度も低下する．更に A^{-1} は A のスパース性を全く保存しないという致命的な欠点をもつことに注意されたい．

さて，上記のような馬鹿げた解法は除外して，線形方程式の解法は，**直接解法** (direct method) と，**間接解法** (indirect method または iterative method) に大別される．直接解法は，**丸め誤差** (round-off error) がないと仮定すれば，有限回の演算で厳密解を求めることが出来る．一方，間接解法は適当な近似解から出発して，逐次近似手法で解を求める手法で，得られる解の精度は演算の手間に依存する．間接解法の代表的なものとしては，**ザイデル法，共役傾斜法，SOR 法**[14,15]などがあるが，これらはいずれも係数行列の性質がかなり良い（例えば，対称かつ正定値）場合しか収束の保障がない．解の精度に対する信頼性，対象とする行列の一般性と演算の手間を有限に保障する点で直接解法が優れており，最近は限られた分野の問題をのぞいて，大規模なスパース行列に対しても直接解法が用いられている．

スパースな行列に対する処理技法といっても——プログラミングの面では種々の工夫を要するが——本質的には従来から知られている行列演算のアルゴリズムを基本としている．そこで本章では，一般行列演算の立場から，行列のスパース性を利用し得る線形方程式の解法および関連したトピックスについて解説する．詳細については文献 15)〜20) などを参照されたい．

2.1 ガウス消去法と消去型逆行列

ガウス消去法 (Gaussian elimination) の第1段階は，$A=\{a_{ij}\}$ を $u_{ij}=1$，且つ $i>j$ のとき $u_{ij}=0$ という構造の行列 $U=\{u_{ij}\}$——**単位上三角行列**(unit upper triangular matrix) と呼ばれる——に変換する段階で，**三角化** (triangularization) と呼ばれる．実際に U を求める手続きは n ステップから成っており，$A^{(1)}=A$，$A^{(n+1)}=U$ とおけば，各ステップの操作は形式的に

$$A^{(k+1)}=L_kA^{(k)};\ k=1,\ 2,\ \cdots,\ n \tag{2.3}$$

と表現できる．ここで $A^{(k)}=\{a_{ij}{}^{(k)}\}$ は k 番目のステップが始まる直前の行列で，**図-1**（a）に示すように $(k-1)$ 列目まで上三角化されている．また，L_k は，図-1（b）に示すように，k 列目の対角項以下 $\eta_i{}^{(k)};\ i=k\sim n$ を除いては単位行列と同じ構造を持つもので，その要素の値は $A^{(k)}$ の k 列目の対角項より下の要素を全て0に変換するように決定される．L_k の要素 $\eta_i{}^{(k)};\ i=k\sim n$ および $A^{(k+1)}$ の k 行目以下の要素の値は次の計算回順によって求められる．

$$\begin{cases}\eta_k{}^{(k)}=1/a_{kk}{}^{(k)}\\ \eta_i{}^{(k)}=-a_{ik}{}^{(k)}\eta_k{}^{(k)};\ i=k+1,\ \cdots,\ n\\ a_{ij}{}^{(k+1)}=a_{ij}{}^{(k)}+\eta_i{}^{(k)}a_{kj}{}^{(k)};\ i=k,\\ \cdots n,\ j=k+1,\ \cdots,\ n\end{cases} \tag{2.4}$$

上記の手順によって $U=A^{(n+1)}$ を求めるまでに丁度

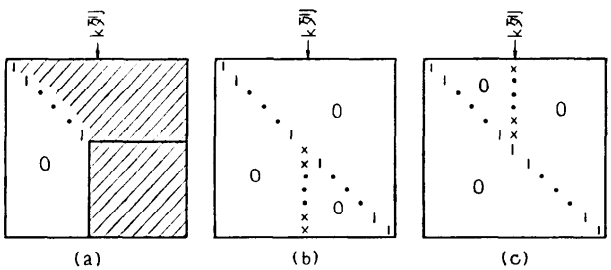

図-1 ガウス消去法と消去型逆行列
—$A^{(k)}(a)$，$L_k(b)$ および $U_k(c)$

n 回の除算が行われるが，この除数 $a_{kk}^{(k)}$ を k ステップ目の**ピボット** (pivot) という．

ガウス消去法は第2段階は与えられた右辺 $\boldsymbol{b}$ と，三角化によって求められた L_k; $k=1\sim n$ から

$$\boldsymbol{y}=L_n\cdots L_2L_1\boldsymbol{b} \tag{2.5}$$

で与えられるベクトル $\boldsymbol{y}$ を求めることで，**前進代入** (forward substitution) と呼ばれる．実際に $\boldsymbol{y}$ を求める計算は，例えば次の手順で行われる．

$$\left.\begin{cases} c_1=b_1,\ y_1=\eta_1^{(1)}c_1 \\ c_i=b_i+\sum_{k=1}^{i-1}\eta_i^{(k)}c_k \\ y_i=\eta_i^{(i)}c_i \end{cases}\right\};\ i=2,\ \cdots,\ n \tag{2.6}$$

式(2.3)，および $A^{(1)}=A$，$A^{(n+1)}=U$ に着目すれば，三角化によって A は

$$L_n\cdots L_2L_1A=U \tag{2.7}$$

と変換されたことになる．更に前進代入の結果 (2.5) を組み合わせれば，元の方程式 (2.1) は

$$U\boldsymbol{x}=\boldsymbol{y} \tag{2.8}$$

という上三角行列を係数とする方程式に変換されたことになる．ガウス消去法の第3段階はこれから所要の解

$$\boldsymbol{x}=U^{-1}\boldsymbol{y} \tag{2.4}$$

を求めることで，**後退代入** (backward substitution) と呼ばれる．実際に $\boldsymbol{x}$ を求める操作は次の手順で行われる．

$$\begin{cases} x_n=y_n \\ x_i=y_i-\sum_{k=i+1}^{n}u_{ik}y_k;\ i=n-1,\ \cdots,\ 1 \end{cases} \tag{2.10}$$

行列 A が完全に密と仮定して，ガウス消去法の演算の手間を見積ると，先ず三角化において n 回の除算と，約 $n^3/3$ 回の乗算及び加減算が必要とされる．また，前進代入と後退代入の各々で約 $n^2/2$ 回の乗算及び加減算が行われる．ガウス消去法と後述のクラウト分解法は線形方程式を解くための最も能率の良い*方法とされており，行列のスパース性を利用した大規模システムの解析プログラムの殆んどはこのいずれかを利用している．

さて，Uが単位上三角行列であることに着目すれば

$$U_2U_3\cdots U_nU=I_n \tag{2.11}$$

という関係を満たすことが容易に確められる．ここで I_n は n 次の単位行列，U_k は図-1 (c) に示すように，k 列目の対角項より上 $\xi_i^{(k)}$; $i=1\sim k-1$ を除いては単位行列と同じ構造を持つ行列である．U_k; $k=2\sim n$ の各要素 $\xi_i^{(k)}$; $i=1\sim k-1$ の値は

$$\xi_i^{(k)}=-u_{ik};\ i=1,\ 2,\ \cdots,\ k-1 \tag{2.12}$$

によって与えられる．式(2.7) と (2.11) を組み合わせれば

$$A^{-1}=U_2\cdots U_nL_n\cdots L_2L_1 \tag{2.13}$$

が得られ，この形式で表わされた逆行列を**消去型逆行列** (elimination form of inverse) という．消去型逆行列は A^{-1} を陽に与えてはいないが，あるベクトル $\boldsymbol{b}$ に対しての $A^{-1}\boldsymbol{b}$ の値は，A^{-1} が陽に与えられた時と同じように n^2 回の乗算と加減算——実際には式 (2.6)，(2.10) のような手順による——によって求められる．実際に (2.13) の右辺を計算機のメモリー中に記憶するには，各 U_k の k 列目の対角項より上の要素と各 L_k の対角項以下の要素だけを覚えておけばよい．一般に A がスパースでも A^{-1} は完全に密になるが，消去型逆行列の形で表現すれば A のスパース性がかなり保存されるという利点がある．消去型逆行列は，後述する積型逆行列よりも演算量，精度，スパース性の保存という面で優れており[22]，最近は積型逆行列の代りに消去型逆行列を利用した線形計画法のプログラムも出現している[31]．

2.2 クラウト分解法

ガウス消去法における三角化の結果(2.7) において

$$L=L_1^{-1}L_2^{-1}\cdots L_n^{-1} \tag{2.14}$$

と書けば，L は下三角行列になることは明らかである．従って (2.7)，(2.14) より A は下三角行列 L と単位上三角行列 U の積

$$A=LU \tag{2.15}$$

に分解されることになる．**クラウト分解法** (Crout factorization) は (2.15) の分解を直接求める方法である．$L=\{l_{ij}\}$ の対角項以下 ($i\geqq j$) および $U=\{u_{ij}\}$ の対角項より上の要素の値は，$k=1,\ 2,\ \cdots,\ n$ について次の計算をすることによって求められる．

$$\begin{cases} l_{ik}=a_{ik}-\sum_{m=1}^{k-1}l_{im}u_{mk};\ i=k,\ \cdots,\ n \\ u_{kj}=\left(a_{kj}-\sum_{m=1}^{k-1}l_{km}u_{mj}\right)\Big/l_{kk};\ j=k+1,\ \cdots,\ n \end{cases} \tag{2.16}$$

この計算を行う過程を**三角分解** (triangular factorization) という．式 (2.1) の解 $\boldsymbol{x}$ を求めるための前進代入，後退代入の計算はガウス消去法とほぼ同じである．

式 (2.14) より，ガウス消去法における各 L_k; k

---

* 行列が密な場合には，これより理論的に速い方法[21]も知られているが，行列のスパース性を利用できないという意味で実用的価値はない．

$=1\sim n$ の要素 $\eta_i^{(k)}$; $i=k\sim n$ とクラウト分解法における $L=\{l_{ij}\}$ の要素は次式によって関係付けられる.

$$\begin{cases}\eta_k^{(k)}=1/l_{kk}\\ \eta_i^{(k)}=-l_{ik}\eta_k^{(k)};\ i=k+1\sim n\end{cases}\qquad(2.17)$$

上の関係は,ガウス消去法とクラウト分解法の計算手順は異なるが,元の行列 A が変換される形は本質的に同じであることを示している.A の消去型逆行列がクラウト分解法によっても得られることはいうまでもない.

クラウト分解法の特徴は,ガウス消去法における(2.3)の計算の代りに(2.16)の内積計算を行うことにあり,積和を順次,レジスタ中に作っていくことができる.積和の計算を高精度演算でレジスタ中に作っていく方法を**内積累和法** (inner product accumulation)[20]といい,累積誤差の少い結果を得ることができる.ガウス消去法とクラウト分解法を比べた場合,総演算量は同じであるがプログラムという面からみると,クラウト分解法の方が精度がよいだけでなく,STORE 及び LOAD 命令の数がガウス消去法の場合よりはるかに少なくてすむ.したがって,一般にはクラウト分解法の方が優れているが,スパース行列処理ではデータ構造とそのアクセス手法,ピボット選択(**2.5** 節参照)の容易さなどを考慮して,問題に適したアルゴリズムの選択が行われている.

クラウト分解法の変形として,式(2.5)の L を単位下三角行列(代りに U の対角項は1とは限らない)となるように分解する方法や,L, U 両方の対角項が1となるように対角行列 D を導入して,

$$A=LDU\qquad(2.18)$$

と分解する方法などがあるが,いずれも式(2.16)の型の内積計算を基本としており,本質的な違いはない[15].

2.3 対称行列の三角化法

回帰分析,構造解析,偏微分方程式の数値解法など多くの分野で対称行列(かつ正定値)を係数行列とする大規模な連立方程式を解く問題があらわれる.そのスパース行列処理においてはその対称性をうまく利用することにより,一般の行列の場合よりもはるかに処理時間とメモリーの節約をはかることができる.ここでは対称行列の三角分解の代表的なアルゴリズムである**コレスキー** (Cholesky) 法について説明する.

行列 A が対称な場合,$A=U^TU$ (U^T は U の転置行列,以下 T は転置を意味するものとする)と三角分解できる.Uは上三角行列である.コレスキー法はこの対称な三角分解を得るための一方法であって,以下のように計算を行う.

$$\begin{cases}u_{11}=\sqrt{a_{11}}\\ u_{1j}=a_{1j}/u_{11};\ j=2,\ \cdots,\ n\\ u_{ii}=\left(a_{ii}-\sum_{k=1}^{i-1}u_{ki}\right)^{1/2};\ j=2,\ \cdots,\ n\\ u_{ij}=\left(a_{ij}-\sum_{k=1}^{i-1}u_{ki}u_{kj}\right)\Big/u_{ii};\\ i=2,\ \cdots,\ n;\ j=i+1,\ \cdots,\ n\end{cases}\qquad(2.19)$$

コレスキー法は対角項の計算で平方根の計算が行われることに特徴がある.密な行列の場合について,ガウス法など一般の行列を解くアルゴリズムと計算の手間とメモリー量を比較すると,ほぼ半分ですむことがわかる.行列が対称かつ正定値の場合は,どのような順序でピボットをとっても数値的に安定(丸め誤差が少い)であることが保障されている[18].

2.4 ガウス・ジョルダン法と積型逆行列

前述のガウス消去法が係数行列 A の三角化に基本をおいているのに対して,**ガウス・ジョルダン法**(Gauss-Jordan elimination) は,A を対角行列に変換する操作と考えられる.アルゴリズムは n ステップから成っており,$A^{(1)}=A$, $A^{(n+1)}=I_n$ とおけば,各ステップの操作は形式的に

$$A^{(k+1)}=T_kA^{(k)};\ k=1,\ 2,\ \cdots,\ n\qquad(2.20)$$

と表現できる.ここで $A^{(k)}=\{a_{ij}^{(k)}\}$ は k ステップ目が始まる直前の行列で,**図-2**(a)に示すように,$(k-1)$ 列目まで単位行列と同じである.また T_k は図-2(b)に示すように,k 列目以外は単位行列と同じ構造を持つもので,その要素の値は $A^{(k)}$ の k 列目の非対角項を0に変換するように決定される.T_k の要素 $\zeta_i^{(k)}$; $i=1\sim n$ および $A^{(k+1)}$ の要素の値は,以下の計算手順によって決定される.

$$\begin{cases}\zeta_k^{(k)}=1/a_{kk}^{(k)}\\ \zeta_i^{(k)}=-a_{ik}\zeta_k^{(k)};\ i=1,\ \cdots,\ k-1,\ k+1,\ \cdots,\ n\\ a_{ij}^{(k+1)}=a_{ij}^{(k)}+\zeta_i^{(k)}a_{kj}^{(k)};\ i=1,\ \cdots,\ n;\\ j=k+1,\ \cdots,\ n\end{cases}$$

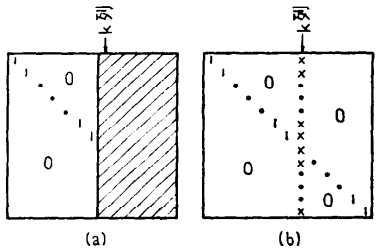

図-2 ガウス・ジョルダン法と積型逆行列
—$A^{(k)}$ (a) と T_k (b)

(2.21)

上記の計算における $a_{kk}^{(k)}$ を，ガウス消去法の場合と同じように，k ステップ目のピボットという．ピボットに関する n 回の除算に加えて約 $n^3/2$ 回の乗算と加減算が行われる．

式 (2.20) および $A^{(1)}=A$，$A^{(n+1)}=I_n$ に着目すれば A は上記の手続きによって，

$$T_n\cdots T_2T_1A=I_n \tag{2.22}$$

と対角化されたことになる．従って A の逆行列は，

$$A^{-1}=T_n\cdots T_2T_1 \tag{2.23}$$

で表わされる．この表現を A の**積型逆行列** (product form of inverse) という．実際に式 (2.23) の右辺を計算機のメモリーに記憶するには，各 T_k の k 列目の要素だけを覚えておけばよい．

積型逆行列と消去型逆行列の要素の間にも密接な関係がある．特に，各 T_k の対角項以下の要素は，対応する L_k の要素に等しい，即ち

$$\zeta_i^{(k)}=\eta_i^{(k)};\ i\geqq k \tag{2.24}$$

であることが知られている[13]．しかし T_k の対角項より上の部分は，通常 U_k よりも多くの非0要素を有し，スパース性の保存という面で積型逆行列は消去型逆行列よりも劣っている．

さて，積型逆行列 (2.23) から解

$$\boldsymbol{x}=T_n\cdots T_2T_1\boldsymbol{b} \tag{2.25}$$

を求める——これを**代入操作** (substitution) という——には，$\boldsymbol{b}^{(1)}=\boldsymbol{b}$，$\boldsymbol{b}^{(n+1)}=\boldsymbol{x}$ とおいて，$k=1, 2, \cdots, n$ について次の計算を行えばよい．

$$\begin{cases} b_k^{(k+1)}=\zeta_k^{(k)}\cdot b_k^{(k)} \\ b_i^{(k+1)}=b_i^{(k)}+\zeta_i^{(k)}b_k^{(k)};\ i=1, 2, \cdots, k-1, k+1, \cdots, n \end{cases} \tag{2.26}$$

代入操作の演算回数（乗算と加減算）はガウス消去法などと同じように，約 n^2 回である．

積型逆行列は消去型逆行列と比べて演算回数，精度，スパース性の保存の面で劣っているので，方程式を解く問題に用いるのは得策でない．しかし，シンプレクス法の反復操作に必要な情報を完全に保存できるという利点を持っているので，線形計画法のプログラムにおいて広く利用されている[23-25]．

2.5 修正行列に対する方法

実際の応用では，係数行列の一部の要素が順次変更されて出来る一連の線形方程式を解くことを必要とする場合がある．特に (2.1) の解析結果をもとに

$$\hat{A}=A+\boldsymbol{c}\boldsymbol{r}^T \tag{2.27}$$

を係数とする方程式

$$\hat{A}\boldsymbol{x}=\boldsymbol{b} \tag{2.28}$$

を解く問題は応用上重要である．ここで $\boldsymbol{c}, \boldsymbol{r}$ は n 次の列ベクトルとする*．

ハウスホルダーの公式[16]によれば，$\hat{A}$ の逆行列は

$$\begin{cases} \hat{A}^{-1}=(I_n-\lambda A^{-1}\boldsymbol{c}\boldsymbol{r}^T)A^{-1}; \\ \lambda=1/(1+\boldsymbol{r}^TA^{-1}\boldsymbol{c}) \end{cases} \tag{2.29}$$

によって与えられる．A の積型，または消去型逆行列がすでに求まっているとして，実際の (2.29) の計算は次の手順で行われる．

$$\begin{cases} ① \quad \boldsymbol{y}=A^{-1}\boldsymbol{c} \\ ② \quad \lambda=1/(1+\boldsymbol{r}^T\boldsymbol{y}) \\ ③ \quad \boldsymbol{z}=\lambda\boldsymbol{y} \end{cases} \tag{2.30}$$

これによって，修正された行列の逆行列は

$$\hat{A}^{-1}=(I_n-\boldsymbol{z}\boldsymbol{r}^T)A^{-1} \tag{2.31}$$

という形で表現されたことになる．計算機メモリーの中にはあらたに $\boldsymbol{z}$ と $\boldsymbol{r}$ だけを記憶しておく．更に，(2.1) の解 $\boldsymbol{x}=A^{-1}\boldsymbol{b}$ が与えられているとして，(2.28) の解を求めるには次の計算を行えばよい．

$$\begin{cases} ④ \quad \mu=\boldsymbol{r}^T\boldsymbol{x} \\ ⑤ \quad \hat{\boldsymbol{x}}=\boldsymbol{x}-\mu\boldsymbol{z} \end{cases} \tag{2.32}$$

この公式を用いることは，$\hat{A}$ の消去型あるいは積型逆行列をあらたに求めることに比べればはるかに有利である．

(2.27) の特殊な場合として，A の k 列目 $\boldsymbol{a}^{(k)}$ が $\hat{\boldsymbol{a}}^{(k)}$ で置き代えられたとき，すなわち

$$\boldsymbol{c}=\hat{\boldsymbol{a}}^{(k)}-\boldsymbol{a}^{(k)} \tag{2.33}$$

且つ $r_k=1$，$r_i=0$ $(i\neq k)$ の場合には，もっと能率の良い $\hat{A}^{-1}$ の表現法が知られている．A の消去法逆行列 (2.13) が与えられているとすれば，$\hat{A}$ の逆行列は

$$\hat{A}^{-1}=U_2\cdots U_{k-1}\hat{T}_kU_{k+1}\cdots U_nL_n\cdots L_1 \tag{2.34}$$

で与えられる．ここで $\hat{T}_k$ は積型逆行列における T_k と同じ，すなわち図-2（b）の構造を持つ行列で，その k 列目の要素 $\hat{\zeta}_i^{(k)}$；$i=1\sim n$ は

$$\begin{cases} \hat{\zeta}_k^{(k)}=1/\alpha_k \\ \hat{\zeta}_i^{(k)}=-\alpha_i\hat{\zeta}_k^{(k)} \end{cases} \tag{2.35}$$

によって決定される．また，ベクトル $\alpha=(\alpha_1\cdots\alpha_n)^T$ は消去型逆行列の一部を用いた代入操作

$$\alpha=U_{k+1}\cdots U_nL_n\cdots L_1\hat{\boldsymbol{a}}^{(k)} \tag{2.36}$$

によって求められる．(2.34) による $\hat{A}$ の逆行列表現は (2.31) によるものに比べて，必要な計算手数とスパース性保存の両面で優れている．

係数行列を何回も変化させる時は，上記のような処

---

* $n\times n$ の行列 $\boldsymbol{c}\boldsymbol{r}^T$ の階数が1であることに注意されたい．

理を繰り返すことになるが，この場合丸め誤差が累積するので，適当な時期に新しい行列についてのガウス消去などをやり直した方が良い．

係数行列の一部の列が変化した場合の線形方程式の解法は，線形計画法に関連した重要な問題である．ここでは基本的な方法を二つ紹介したが，これらの詳細やもっと新しい方法については文献 26)～31) などを参照されたい．

2.6 ピボットの選択

これまでの線形方程式解析手法の説明においては，各ステップにおいて，未処理の部分の一番上の対角項をピボットに取ると仮定して来た．例えばガウス消去の k ステップにおいて $a_{kk}^{(k)}$ をピボットに取ると仮定した．しかし，係数行列 A が正則であっても，$a_{kk}^{(k)}=0$ ならば (2.4) の計算は実行不可能となる．また，実際の数値計算は有限桁の浮動小数点演算によって行われるので，ピボットが0でなくても値が小さすぎると，丸め誤差の影響により信頼できる解が得られない．従って高精度の解を得るためには，なるべく絶対値の大きい要素をピボットに選ぶ必要がある[8]．このようなピボット選択操作を**絶対値によるピボティング**(pivoting for size) という．最も広く用いられているのは，k ステップ目において，k 列目の最大の要素

$$|a_{ik}^{(k)}|=\max_i|a_{ik}^{(k)}|\ ;\ k\leqq i\leqq n \qquad (2.37)$$

をピボットに選ぶ方法である．これを**部分的ピボティング** (partial pivoting) という．この場合，k ステップ目の計算 (2.4) の前に k 行目と s 行目の入れ換えが行われる．更に精度を上げるために——手数は増加するが——$A^{(k)}$ の未処理の部分全ての中から最大の要素

$$|a_{st}^{(k)}|=\max_{i,j}|a_{ij}^{(k)}|\ ;\ k\leqq i,\ j\leqq n \qquad (2.38)$$

を選ぶ方法もある．これを**完全ピボティング**(complete pivoting) という．この場合，行と列の両方の入れ換えが必要である．

絶対値によるピボティングを行えば，丸め誤差による精度の低下をある程度防ぐことができる．しかし，係数行列によってはこれを行っても効果のない場合もある．一般に (2.1) を解く際に丸め誤差の影響を受け易いか否かは A の性質によって決まり，その尺度は

$$\mathbf{cond}\,A=\|A\|\cdot\|A^{-1}\| \qquad (2.39)$$

で与えられる[18]．この数を行列の**条件数** (condition number) という．条件数が大きい場合は，その行列は**たちが悪い** (ill-conditioned) といわれ，精度のよい解を得るためには浮動小数点演算の桁数を増やす必要がある．

ピボットの選択について，誤差の立場から述べて来たが，大規模なスパース行列を取り扱う場合には，新しく発生する非0要素の数を少くするようにピボットを選ぶことが計算時間短縮のために重要である．このようなピボットの選び方については第3，4章に述べる．さて行列のスパース性を保存するようにピボットを選ぶと，前述の意味で精度が低下することがしばしば起る．そこでピボットの値が小さすぎる時は，これに適当な定数を加えたものによってピボット選択を行い，最後に得られた解を修正する方法 (**2.7** 参照) や，精度の低い解を反復計算によって改良する方法(**2.8** 参照) などが補助的に用いられている．

何らかのピボット選択基準によってもとの係数行列 A の行や列の入れ換えが行われた場合の解を表現するには，次に定義する行列 P_{ij} を導入すると便利である．P_{ij} はその i,j 番目の行と列だけから成る 2×2 の部分が

$$\begin{array}{c} \\ i \\ j \end{array}\begin{array}{c} \begin{array}{cc} i & j \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array}$$

となっている他は単位行列と同じものであると定義する．これを用いれば，A の i,j 行を入れ換えた行列は $P_{ij}A$ によって，i,j 列を入れ換えた行列は AP_{ij} によって与えられる．もっと一般的に，整数 $1\sim n$ から成る順列 $\alpha=(\alpha_1,\alpha_2\cdots,\alpha_n)$ に対して，**置換行列**(permutation matrix) P_α は単位行列 I_n を α_1 行，α_2 行……という順に並べ換えたものと定義する．これを用いれば $P_\alpha A$(AP_α) は A を α_1 行(列)，α_2 行(列)…という順に並べ直したものを表わすことになる．

(2.1) の解析において，行や列の置換が行われた時は，一般に二つの置換行列 P, Q に対して

$$\bar{A}=PAQ^T \qquad (2.40)$$

を係数行列とみなした処理が行われたことになる．

そこで代入操作によって解を求める時は，実際には

$$\boldsymbol{x}=Q^T\bar{A}^{-1}P\boldsymbol{b} \qquad (2.41)$$

という計算を行わなければならない．ここで (2.41) の導入に当って，置換行列 P の性質

$$P^{-1}=P^T \qquad (2.42)$$

が使われたことに注意されたい．実際に P, Q に関する情報を記憶するには，二つの整数ベクトルを用意すれば良い．

2.7 ピボット修正法

係数行列のスパース構造が一定で，非0要素の値が異なる一連の線形方程式を解く場合，ピボットにとる要素とその処理順序を——スパース性を最大限保存するように——あらかじめ決定しておくのが得策である．すなわち，あらかじめ係数行列の行と列の適当な置換を行って，実際の数値解析では対角項を上から順番にピボットとして選ぶことに相当する．ここに述べる**ピボット修正法** (modifying pivot elements) は，次にとるべきピボットの値が小さくなるとき，適当な大きさの値をピボット要素に加えて修正を行い，修正された行列に対する線形方程式を解き，最後に修正分を補正してもとの方程式の解を得る方法[32,33]である．この方法の利点は，ピボットの順序を変更しないで丸め誤差を制御できることにあり，スパース行列処理に向いた方法といえる．

方程式 (2.1) の解析，例えばガウス消去において，第 k ピボット $a_{kk}^{(k)}$ の絶対値が非常に小さいとき，これに適当な数 g_k を加えてから残りの計算を実行したとする．これによって得られる解は，A の代りに

$$\hat{A}=A+g_k\boldsymbol{e}^{(k)}\boldsymbol{e}^{(k)T} \tag{2.43}$$

を係数とする方程式

$$\hat{A}\hat{\boldsymbol{x}}=\boldsymbol{b} \tag{2.44}$$

の解ということになる．ここで $\boldsymbol{e}^{(k)}$ は第 k 要素が1で他は全て0なる列ベクトルである．ここでハウスホルダーの公式を適用*すれば

$$\begin{cases}A^{-1}=(I_n-\lambda\boldsymbol{w}\boldsymbol{e}^{(k)T})\hat{A}^{-1}; \\ \lambda=g_k/(w_kg_k-1)\end{cases} \tag{2.45}$$

という関係が得られる．ここで $\boldsymbol{w}$ は

$$\hat{A}\boldsymbol{w}=\boldsymbol{e}^{(k)} \tag{2.46}$$

の解，w_k はその第 k 要素である．従ってもとの方程式の解 $\boldsymbol{x}$ は

$$\boldsymbol{x}=\hat{\boldsymbol{x}}-\mu\boldsymbol{w};\ \mu=\lambda\hat{x}_k \tag{2.47}$$

によって与えられる．ここで $\hat{x}_k$ は $\hat{\boldsymbol{x}}$ の第 k 要素である．このピボット修正法によって増加する計算の手間は $\hat{A}$ の消去型逆行列と $\boldsymbol{e}^{(k)}$ から $\boldsymbol{w}$ を求める代入操作の手間によって，ほぼ見積られる．

2個以上のピボットが修正される場合も，同様の操作をくり返せば良い．ピボット修正法によって得られる解は，十分良い精度を持つことが，丸め誤差解析によって裏付けられている[33]．

2.8 反復改良法

(2.1) の何らかの数値解法による近似解 $\boldsymbol{x}^*$ の精度が不満足の場合には，次のように**反復改良法** (iterative refinement) によって，若干の手間の増加で高精度の解を得ることができる[34]．すなわち $\boldsymbol{x}^{(1)}=\boldsymbol{x}^*$ とおいて $k=1, 2, \cdots$ について

$$\begin{cases}\boldsymbol{r}^{(k)}=\boldsymbol{b}-A\boldsymbol{x}^{(k)} \\ \Delta\boldsymbol{x}^{(k)}=A^{-1}\boldsymbol{r}^{(k)} \\ \boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\Delta\boldsymbol{x}^{(k)}\end{cases} \tag{2.48}$$

という反復計算を行えば，$\boldsymbol{x}^{(k)}$ は k の増加につれて $\boldsymbol{x}$ のより良い近似を与える．但し，残差 $\boldsymbol{r}^{(k)}$ は高精度計算で求めることが重要である．

反復計算1回当りの手間は，行列が密としてほぼ $2n^2$，すなわち方程式を最初に解く手間の $6/n$ 程度である．通常，数回の反復計算を行えば十分である．収束の速度は計算機の有効桁数と行列 A の条件数の関係できまる．いま，計算機の浮動小数点数の底を β，有効桁数を t とすると，必要な反復回数 m はほぼ

$$m=t/\log_\beta(\|\boldsymbol{x}^{(1)}\|/n\|\Delta\boldsymbol{x}^{(1)}\|) \tag{2.49}$$

で与えられる．また，

$$\log_\beta(\mathbf{cond}\,A)>t \tag{2.50}$$

の場合，反復改良法の効果はない[20]．

2.9 固有値問題

行列の固有値問題に対しても，最近のスパース性を利用して計算時間，メモリーを節約する試みがなされるようになってきた[35,36]．行列の固有値，固有ベクトルを求めるには，先ず与えられた行列を相似変換によって**ヘッセンバーグ行列****に（行列が対称なら直交変換によって**三項行列***に）変換してから反復計算を行うのが良いとされている[18,19]．この相似変換において，元の行列のスパース性を如何に保存するかがスパース行列処理における問題である．固有値問題に関するスパース処理は複雑な割には応用分野が限られているので本講座では省略することにする．

* 2.5 節とは A, $\boldsymbol{x}$ と $\hat{A}$, $\hat{\boldsymbol{x}}$ の立場が逆になっていることに注意されたい．

** 正方行列 $A=\{a_{ij}\}$ において $a_{ij}=0$; $i\geq j+2$ ならば A はヘッセンバーグ行列と呼ばれる．A, A^T 共にヘッセンバーグ行列ならば，A は三項行列と呼ばれる．

参 考 文 献

1) G.B. Dantzig & W. Orchard-Hays: The Product Form of Inverse in the Simplex Method, Math. Comp., Vol. 8, pp. 64～67 (1954).
2) H.M. Markowitz: The Elimination Form of the Inverse and its Application to Linear Programming, Man. Sci., Vol. 3, pp. 255～269 (1957).

3) R. Courant & D. Hilbert: Methods of Mathematical Physics (Vol. 1), Interscience, New York (1953).
4) D.M. Young: Iterative Methods for Solving Partial Difference Equations of Elliptic Type, Trans. Amer. Math. Soc., Vol. 76, pp. 92～111 (1954).
5) D. W. Peaceman & H. H. Rachford, Jr.: The Numerical Solution of Parabolic and Elliptic Differential Equations, J. SIAM, Vol. 3, pp. 28～41 (1955).
6) L. J. Arms, L. D. Gates & B. Zondek: A Method of Blok Iteration, J. SIAM, Vol. 4, pp. 220～229 (1956).
7) E.H. Cuthill & R.S. Varga: A Method of Normalized Block Iteration, J. ACM, Vol. 6, pp. 236～244 (1959).
8) R.E. Langer (ed.): Boundary Problems in Differential Equations, Univ, of Wisconsin Press, Madison (1959).
9) S.V. Parter: On 'Two-Line' Iterative Methods for the Laplace and Biharmonic Difference Equations, Numer. Math., Vol. 1, pp. 240～252 (1959).
10) R.A. Willoughby (ed.): Sparse Matrix Proceedings, Rep. No. RA 1 (11707), IBM, Yorktown Heights (1969).
11) J.K. Reid (ed.): Large Sparse Sets of Linear Equations, Academic Press, London, Proc. Oxford Conf. (1971).
12) D.J. Rose & R.A. Willoughby (eds.): Sparse Matrices and Their Applications. Plenum Press, New York, Proc. IBM Conf. (1972).
13) R.P. Tewarson: Sparse Matrices, Academic Press, New York (1973).
14) R.S. Varga: Matix Iterative Analysis, Prentice-Hall, Englewood Cliffs (1962).
15) J.R. Westlake: A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, John Wiley & Sons, New York (1968).
16) A.S. Householder: Principles of Numerical Analysis, McGraw-Hill, New York (1953).
17) A.S. Householder: The Theory of Matrices in Numerical Analysis, Blaisdell, New York (1964).
18) J.H. Wilkinson: The Algebraic Eigenvalue Problem, Oxford Univ. Press, London and New York (1965).
19) L. Fox: Introduction to Numerical Linear Algebra, Oxford Univ. Press, London and New York (1965).
20) G.E. Forsythe & C. B. Moler: Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs (1967).
21) V. Strassen: Gaussian Elimination is not Optimal, Numer. Math., Vol. 13, pp. 354～356 (1969).
22) G.B. Dantzig ***et al.***: Sparse Matrix Techniques in Two Mathematical Programming Codes, in 10), pp. 85～99 (1969).
23) G. B. Dantzig: Linear Programming and Extensions, Princeton Univ. Press, Princeton (1963).
24) D.M. Smith: Data Logistics for Matrix Inversion, in 10), pp. 127～137 (1969).
25) J. de Buchet: How to Take into Account the Low Density of Matrices to Design a Mathematical Programming Package, in 11) pp. 211～218 (1971).
26) J. P. Roth: An Application of Algebraic Topology—Kron's Method of Tearing—, Quart. Appl. Math., Vol. 17, pp. 1～24 (1959).
27) J.M. Bennett: Triangular Factors of Modified Matrices, Numer. Math., Vol. 7, pp. 217～221 (1965).
28) R.H. Bartels & G.H. Golub: The Simplex Method of Linear Programming Using LU Decomposition, Comm. ACM, Vol. 12, pp. 266～268 (1969).
29) Brayton ***et al***: Some Results on Sparse Matrices, Rep. No. RC 2332, IBM, Yorktown Heights (1969).
30) T. Fujisawa, E.S. Kuh & T. Ohtsuki: A Sparse Matrix Method for Analysis of Piecewise-Linear Resistive Networks, IEEE Trans., Vol. CT-19, pp. 571～584 (1972).
31) J.J.H. Forrest & J.A. Tomlin: Updating Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method, Math. Programming, Vol. 2, pp. 263～268 (1972).
32) O. Wing: Solution of Sparse Systems of Linear Equations witn Diagonal Modifications to Cntrol Numerical Stability, to appear.
33) G.W. Stewart: Modifying Pivot Elements in Gaussian Elimination, Math. Comp., Vol. 28, pp. 537～542 (1974).
34) C.B. Moler: Iterative Refinement in floating Point, J. ACM, Vol. 14, pp. 316～321 (1967)
35) R.P. Tewarson: On the Transformation of Symmetric Sparse Matrices to the Triple Diagonal Form, Internat. J. Comput. Math., Vol. 2, pp. 247～258 (1970).
36) R. P. Tewarson: On the Reduction of a Sparse Matrix to Hessenberg form, ***ibid***, pp. 283～295 (1970).

(昭和50年9月9日受付)