7K－1

ベクトル計算機向き 大規模疎行列の格納法

斎藤 知哉，亘 紀子
日本電気技術情報システム開発㈱

１．序

有限要素法によって離散化して得られる大規模疎行列を係数行列とする連立方程式、あるいは固有値問題の解法として様々な反復解法が開発されてきた。それらのうち、

1. vectorの内積・和・scalar倍
2. matrixとvectorの掛け算

の基本算法のみから成り立っているものも少なくない。それらの反復解法をベクトル計算機上で実現する場合、2.の過程では、それに含まれる乗算、加算の順序が任意であるので様々な可能性が考えられる。

１行目から順に非ゼロ要素を詰めていく方法（行方向詰）は、スカラー計算機では最も自然である。しかしこの方法では、ベクトルとの掛け算は行毎に行うことになり、ベクトル長は通常一ないし二桁程しか期待できず、ベクトル計算機の機能を活かすことが出来ない。

さて、前回我々は同様な題で３種の方法(type1,2,3)を提案したが[1]、それらを改良し高速化することができたので、それについて報告する。我々の数値実験では、最高、行方向詰の２０分の１程度に計算時間を短縮できた。

次節以降は、このうち最も有望と思われるtype1',4について詳しく記述する。各格納法に対応する行列とベクトルとのかけ算の非対角の部分を擬似コードで書くと以下のように書ける。

type 1,1',1" の場合

((k++, $y_i=y_i+a_k*x_{j(k)}$, for i=1,ie$_\ell$),for ℓ =1,nℓ)

type 3,3',4 の場合

((k++, $y_i=y_i+a_k*x_{j(k)}$, $y_{j(k)}=y_{j(k)}+a_k*x_i$,for i=1,ie$_\ell$),for ℓ=1,nℓ)

type1 系 以外は対称行列の為のもので、最深DOループ内でリストベクトルｊに重なりが無い事がベクトル化の必要条件である。

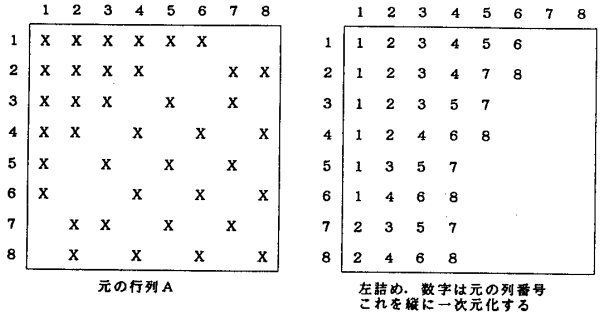

図１

２．新しい行列格納法

（１）タイプ１'

前回発表したタイプ１の格納法（図１）では、同一列中に同じ列番号が並び易い。例えば Ａ＊𝔵 を実行する際に図１に於いて、１回目のループでx_1を６回、２回目のループでx_2を４回、３回目のループではx_3に３回、連続して参照をしている。このようなアクセスをリストベクトルで行うと、更新はないので不正な結果を生ずる事はないが、バンクコンフリクトを引き起こす事がわかった。これを回避するために、図２のようにタイプ１の格納法を改訂する。この変更により、図の例では Ａ＊𝔵 計算時に連続して𝔵の同じ要素にアクセスする回数をタイプ１の６回からタイプ１'の３回に減少させる事が出来る。

（２）タイプ４

前回発表したタイプ３は、対称行列の上三角のみの格納であったが、高速ソートが出来ないと判断し一行にある要素の数を降順に並べなかった。またリストによるアクセスを一方向のみにするため、左詰めしたときに出来る間隙にゼロ要素を詰めていた（図３）。今回は行列の非ゼロ要素の総数に比例する演算量で、ソートした。さらに、ベクトル化時の同一要素二重定義を避ける為の手だてとして、①左列から順に下から走査してゆき、同一の列番号が現れたら、同一行中右にあるもので列の中で一度も使われていない番号と取り替える。そのようなものがないときは、現れた回数の最も少ないものを選ぶ。我々の実験で、同一縦列中の列番号の多重度の最高値は通常２、最高で３であった。②（２度以上現れる要素の列番号）＝（元の列番号）＋（行列のサイズ）＊（同一番号の出現回数－１）をとる。 𝔶＝Ａ＊𝔵 の計算をするとき、ベクトル𝔵及び𝔶は、長さＮの２倍（または３倍）とっておき、例えば出現回数が２回の時は、２度出現する列番号 i について前処理 $\{y_{i+N}=0, X_{i+N}=X_i\}$ 後処理 $\{Y_i=Y_i+Y_{i+N}\}$ をほどこす。対称行列の場合、タイプ１'による格納は、タイプ４による格納の約２倍領域を必要とする。

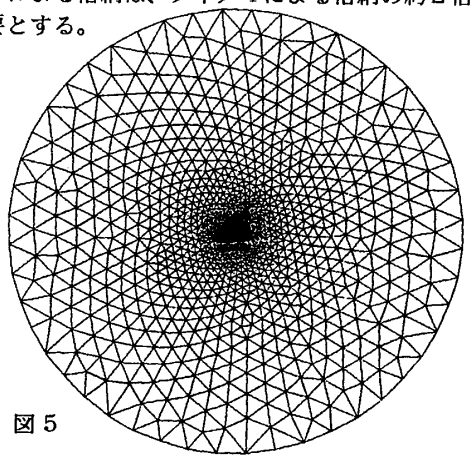

図５

New Storage Schemes of Large-Scale Sparse Matrices For Vectorcomputers
Tomoya Saito,Noriko Watari
NEC Scientific Information System Development,Ltd.

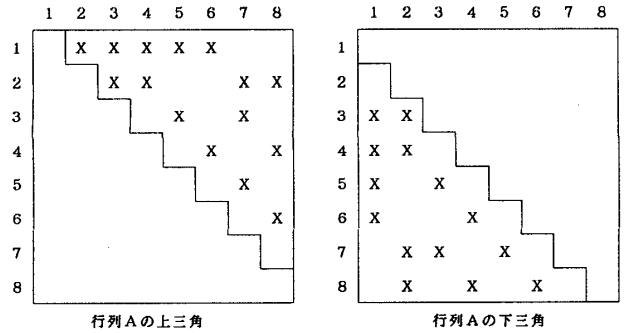

図2

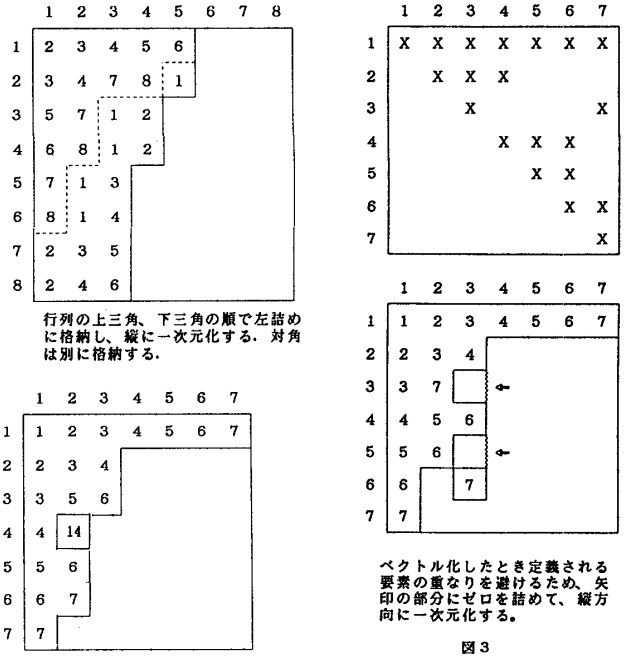

図3

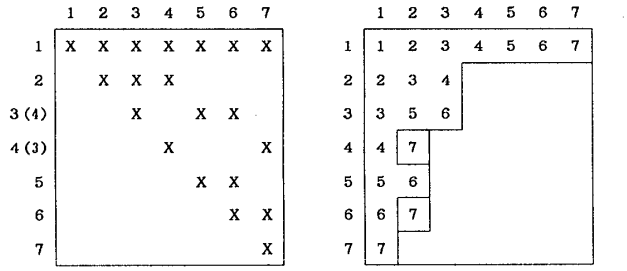

図4

3．数値実験　Ⅰ

考案したスパース行列格納法の有効性を見るために、単位円領域を疎密のある三角形分割（図５）で離散化して得られる有限要素法マトリクス（基底関数１，２，３，４次）を使って実際に時間を測定した。計算機はＮＥＣスーパーコンピュータＳＸ－２を使用した。結果を図６に示す。縦軸が１非ゼロ要素当たりの計算時間、横軸が基底関数の次数である。今回のタイプ１’とタイプ４は、対角方向リスト[2]に比べて３．１～２．３倍、前回提案したタイプ３に比べても２．０～１．４倍高速であることがわかる。タイプ１の計算時間が、基底関数を高次にするにつれて非常に悪化するのは同一要素をリストベクトルで連続して参照するために起こるバンクコンフリクトが原因と考えられる。全体的にみて、１’の修飾版（バンクコンフリクトを減らすように行方向の入れ替えをしている）であるタイプ１”が最も速い　（１’から１”への変換にかかるコストは300回程度の、行方向から１’への変換は2～3回程度のかけ算で回収できる）。

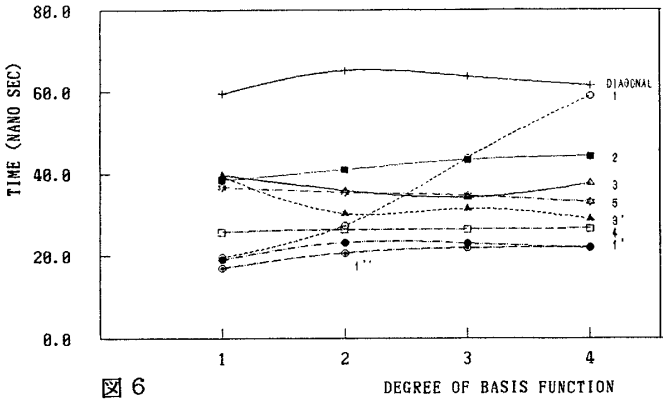

図６

4．　数値実験　Ⅱ

次に具体的問題での数値実験例を挙げる。単位正方形でのLaplacianの最小固有値問題にBradburyらのレイリー商の最小化法[3]を用いた（この方法では１反復あたり、ベクトルの内積が６回、行列・ベクトルのかけ算が２回含まれる）。なお、初期値は乱数を用い、誤差判定用のεは　10^{-9}とした。比較する格納法として、行方向及びタイプ１’を選んだ。基底関数は１次(図で、y11',y1r、前者がタイプ１’)と３次の場合(y31',y3r)について行った。参考として差分法との結果(s)とも比較した（１反復あたりの行列・ベクトルの掛け算回数は差分では半分になる。）。分割数を変化させ、横軸に反復に要した計算時間、縦軸に最終的に得られた固有値と正しい値である$2*\pi^2$の差を取った（図７）。

type1'のものは行方向の１０倍高速である。基底関数が一次のときは５点差分の１０倍の計算コストがかかるが、３次では３００分の１程度で行えることが分かる。係数が場所に依存する場合、境界が長方形から変形した場合等は離散化点のみの情報を使う差分法よりも有限要素法により有利な結果が期待できる。

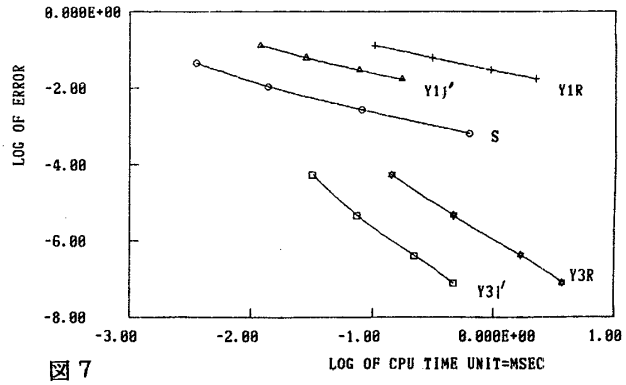

図７

[参考文献]
1)斎藤他：情報処理学会　第39回　全国大会論文集　119-120(1989)
2)速水謙他：情報処理学会　第33回　全国大会論文集　2303-2304(1986)
3)Bradbury,W,W et al:Numerishe Mathematik 9,259-267(1966)