統計数理研究所彙報
第30巻 第2号 (1983)

最小二乗法におけるGivens法とHouseholder法について

電気通信大学 統計数理研究所 田 中 輝 雄

(1982年9月 受付)

1. はじめに

古くから工学・応用数学の分野において現れる大型の有用な行列は，非零成分が少ない行列(疎行列，sparse matrix) として表されることが多い．このような行列に対する解析は密な行列 (dense matrix) を扱う場合とちがって，スパース性を利用することによって計算機の記憶容量・計算時間等を節約することができる．そのために新しいアルゴリズムを考える必要がある [8].

本稿では，スパースな行列が現れる例として，広くデータ解析に用いられている最小二乗問題を扱う．ここでは，現在あまり用いられていない Givens 法と，頻繁に用いられている Householder 法との比較実験を行ない，スパースな問題に対する Givens 法の有効性について述べる．さらに，Givens 法を有効にするための新しいピボット選択について述べる．

2. 線形最小二乗問題

あらゆる分野のデータ解析は，データの検討，モデルの選択・吟味などの複雑な過程をもつ解析であるが，その中心となるのは最小二乗法によるあてはめとなることが多い．最近のように精密なデータが多量に得られるようになると，それらを説明するためのモデルも複雑になり多量のパラメータを推定しなければならなくなってきている．このとき解くべき最小二乗問題は非常に多次元の連立方程式となる．

ここでは線形の場合について考えてみる．つまり，最小二乗問題は行列 A を $(m\times n)$ 行列 rank $(A)=n$ とすれば，

$$S[x]=\|b-Ax\|^2=\sum_{i=1}^{m}\left(b_i-\sum_{j=1}^{n}a_{ij}x_j\right)^2 \tag{1}$$

を最小とするような x を求めることである．ここで，$\|\cdot\|$ はユークリッド・ノルムである．

さて，(1) 式の解を求めるためには，x に関する偏微分を零とおいた

$$A^tAx=A^tb \tag{2}$$

を解けばよい (t は転置の意味)．これは，正規方程式 (normal equation) と呼ばれている．

これを，

$$A^tA=LDL^t \tag{3}$$

(L は下三角行列，D は対角行列) と分解して直接的に解く方法 (Cholesky 分解法) が古くから用いられてきた．しかし，この方法では A^tA の条件数 (誤差の伝播率) がもとの行列 A の条件数の二乗となり，方程式が悪条件 (ill-condition) になりやすい [3]．つまり，計算機内の

有限桁計算では正確に求められない危険が大きくなる.

そこで，行列 A を

$$A = QR \tag{4}$$

と分解する方法が用いられる．ただし，Q は $(m\times m)$ の直交行列，R は $(m\times n)$ の行列で $R=(r_{ij})$ とすると，r_{ij} $(i>j)$ 要素がすべて零の行列である．この (4) 式を (2) 式に代入すると，

$$R^tRx = R^tQ^tb \tag{5}$$

ここで $R=\begin{pmatrix}R_1\\0\end{pmatrix}$, R_1 は $(n\times n)$ の行列，R に対応して $Q^tb=\begin{pmatrix}b_1\\b_2\end{pmatrix}$ とおけば rank $(A)=n$ であるから，

$$R_1x = b_1 \tag{6}$$

となり後退代入により簡単に解を求めることができる．このとき，行列 R_1 は行列 A とノルムの大きさが同じであるから，正規方程式を直接に解く場合とちがって，もとの方程式の条件数を変えず行列の性質が悪くならない．これが，QR 分解法が現在よく用いられている理由である．

この他にもいろいろな解法がある [7]，[9] が，ここでは QR 分解法の中で広く用いられている Householder 法と Givens 法についての比較を行ってみる．Givens 法は密行列に対しては効率が悪いとされてきた [6] が，スパースな問題には有効と思われるからである．

3. QR 分解法

行列 A を QR と分解する方法としては，改良 Gram-Schmidt 法，Householder 法，Givens 法などが挙げられる．前二者に比べて，Givens 法はあまり知られていない．QR 分解法はどの方法を用いても最終的には本質的一意に分解される [7] が，その過程においては各変換によって計算量が大きく異なる．

Table 3 に，密行列における Givens 法と Householder 法の記憶容量と計算量（乗除算）を示す．表中の [one-off] は，QR 分解時に R のみを生成し Q を保持しない場合に必要な記憶容量である．一方 [many-off] は，あとで何回も A を QR と分解したものを用いて多数の b を処理するために Q を R とともに保持する場合に必要な記憶容量である．表中の [Givens (1), (2)] のちがいについては後述する．

この表からわかるように，密な行列に対しては，Givens 法は記憶容量，計算量の両面で

Table 3 Storage and operation counts of dense matrix ($M*N$)

	Storage		Operation (mult'ns and divs)	
	one-off	many-off	decomposition	solve
Normal eq. $A^tA=LDL^t$	MN	$MN+\frac{1}{2}N^2$	$\frac{1}{2}MN^2+\frac{1}{6}N^3$	$MN+N^2$
Householder $A=Q_hR_h$	MN	$MN+N$	$MN^2-\frac{1}{3}N^3$	$2MN-\frac{1}{2}N^2$
Givens (1) $A=Q_gR_g$	MN	$2MN-\frac{1}{2}N^2$	$2MN^2-\frac{2}{3}N^3$	$2MN-\frac{1}{2}N^2$
Givens (2) $A=Q_gR_g$	MN	$MN+N$	$2MN^2-\frac{2}{3}N^3$	$6MN-\frac{5}{2}N^2$

Householder 法よりもコストがかかる．しかし，対象とする行列がスパースな行列であると，Givens 法は Householder 法よりスパース性を保つ働きがある．なぜならば，Givens 法は計算の途中で起こる fill-in（零要素が非零要素になること）を比較的おさえる性質があるからである．すなわち密な行列とちがって記憶容量・計算量などのコストのさがり方が Householder 法よりも Givens 法の方が大きい．

これらのようすを解析するために，まず Householder 法と Givens 法のアルゴリズムをみる．

3.1 Householder 法

Householder 法は鏡像変換を基本にした解法である．行列 A の一列ずつをまとめて非零要素を零にしていくので n 回（列の数）の変換で行列 A を上三角行列 R に変換することができる．r 回の変換をした後の行列を $A^{(r)}$ とすれば

$$A^{(r+1)} = Q^{(r)}A^{(r)} \tag{7}$$

によって $A^{(r)}$ が計算される．変換行列 $Q^{(r)}$ は，

$$Q^{(r)} = I - 2w^{(r)}(w^{(r)})^t / \|w^{(r)}\|^2 \tag{8}$$

と書ける．ここで，$w^{(r)}$ は，

$$w^{(r)} = (0, \cdots, 0, a_{r,r}{}^{(r)} + \mathrm{sign}\,(a_{r,r}{}^{(r)}) \cdot s_r, a_{r+1,r}{}^{(r)}, \cdots, a_{m,r}{}^{(r)}), \tag{9}$$

$$s_r{}^2 = \sum_{i=r}^{m} (a_{ir}{}^{(r)})^2 \tag{10}$$

である．

したがって，Householder 法の場合は行列 Q を明示せずに w のままで計算できるので，Q の具体的な形はどこにも必要ない．よって，Q を保持する場合は Q の情報量としては零にすべき非零要素（fill-in によってできる要素も含む）とその位置がわかればよい．つまり，$Q^{(r)}$ の情報は $w_i{}^{(r)}$, $i \geq r$, $w_i \neq 0$ を保持すればよい．

3.2 Givens 法

Givens 法は固有値問題に広く使われている Jacobi 法と原理が同じである．行列 A に対して，二次元の回転に相当する変換を繰り返して行列 A を上三角行列 R にする．つまり，a_{ij}, $i>j$, $a_{ij} \neq 0$ のすべての非零要素に対して（fill-in によって最初の A よりふえる）左上隅から順次に値を零にしていく．

具体的には，非零要素 a_{ik}, $i>k$ を零にする場合の交換行列 $P^{(ik)}$ を次のように構成する．

$$P^{(ik)} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_{ik} & s_{ik} & \cdots\cdots & \cdots k\text{行} \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & -s_{ik} & c_{ik} & \cdots\cdots & \cdots i\text{行} \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \tag{11}$$

ここで，

$$r = \sqrt{a_{kk}{}^2 + a_{ik}{}^2}, \tag{12}$$

(13) $$c_{ik}=a_{kk}/r,\quad s_{ik}=a_{ik}/r$$

と定めて，$P^{(ik)}$ を A の左から作用させることにより，

(14) $$a_{kj}^{\text{new}}=c_{ik}a_{kj}+s_{ik}a_{ij},\quad k\leq j\leq n,$$

(15) $$a_{ij}^{\text{new}}=-s_{ik}a_{kj}+c_{ik}a_{ij}$$

となり，A の k, i 行の k 番目以降のみが変形され，a_{ik} 要素が零になる．(このとき，k, i 行の $k+1$ 番目以降に fill-in の可能性がある．) つまり，

(16) $$A=P^{(1)t}P^{(2)t}\cdots P^{(l)t}R=QR$$

となって QR 分解が完成する (l は非零要素を零にする回数)．あきらかに，Householder 法と同様に Q の具体的な形はどこにも記憶しておく必要はない．$P^{(ik)}$ は c_{ik} と s_{ik} の2つの情報で十分である．よって，Q の情報量としては，$2l$ 必要となる．すなわち，もし行列が密な行列であれば情報の量は Givens 法は Householder 法の2倍必要となる．(これが Table 3 の Givens (1).) しかし，a_{kk} を固定すると，もとの要素の a_{ik} により c_{ik} と s_{ik} を順次再現できるので，もとの行列の非零要素をそのまま記憶することにより記憶量は約半分ですむ．(Table 3 の Givens (2). この場合，c_{ik} と s_{ik} を毎回再現するので右辺を計算するための計算量が多くなってしまう.) ここでは，記憶量をへらすことに重点をおくために Givens (2) を用いる．

4. ピボット選択

Q を作るときに行列 A の行および列の交換を行なうこと（連立方程式解法の用語を用いてピボット選択と呼ぶ）は，解法の数値的な安定性を保つために重要である．しかし，ここでは直交変換が行列 A の性質を悪くしない解法であることから，ピボット選択を行列 A のスパース性を保つということのために用いる．すなわち，変換時に fill-in がなるべく起こらないようにピボットを選ぶ．このことにより，計算量，記憶容量をへらすことができる．

fill-in は有限であるから，それを最小にするようなピボット選択が存在するが，それを見つけることは計算量が非常に大きくなり，あまり有意義でない [8]．そこで，あまり手間をかけずに，ある程度 fill-in を小さくするアルゴリズムを考える．

アルゴリズム ピボット選択

[(k-1) 列まで変換が終っているとする]

ステップ1 k 列から n 列までの中で非零要素数が最小の列を見つけ，その列を k 列と交換する．

ステップ2 新しい k 列の中の非零要素の中で，その各要素の行の非零要素数の最小の行を k 行と交換する．

ステップ3 k 列の k 番目より下の残りの非零要素をその各要素の行の非零要素数の小さい順に Givens 変換で零にしていく．

Fig. 4 に具体例を示す．(6×5) の非零要素数17個のデータ行列に Givens 変換を適用してみる．第一段として，ピボット選択を行なわずに a_{11} 要素をピボットとして第一列の残りの要素を零にすると10個の fill-in が起こる．ピボット選択を行なった場合は第一行と第二行，第一列と第二列が交換され，もとの行列の a_{22} 要素がピボットになり fill-in は5個ですむ．同様にしてピボット選択を行なわない方法と行なう方法のそれぞれの過程を最後まで行なうと，fill-in はそれぞれ12個と6個となり大きな差として現れる．

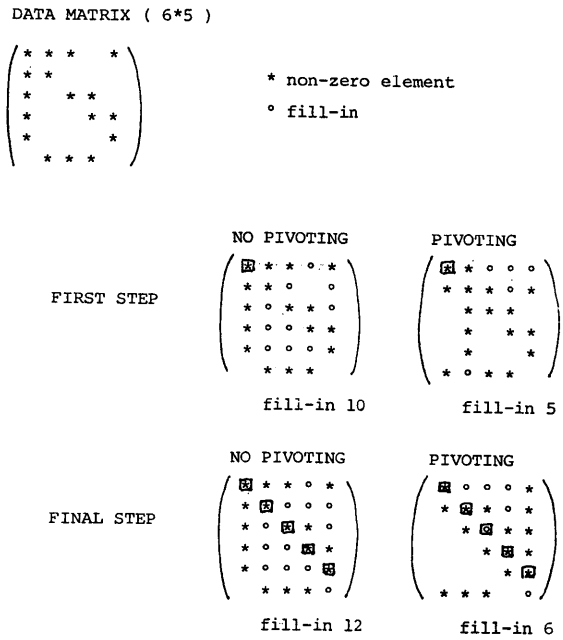

Fig. 4 A comparison of pivoting or not for Givens method

このピボット選択は Householder 法にも用いることができる.

5. データ構造

今回扱っているのはスパースな行列である. よって, 計算機にインプリメントする時に密行列のように二次元の配列を用いるのではほとんど零の要素を持つことになるので無駄である. そこで非零要素のみをたくわえるようなデータ構造を考える. 本稿ではモデル行列の構造は考えない. そこで, 非零要素だけを持ち, fill-in が起こった場合に新しい非零要素が簡単に取りこめるようなダイナミックなデータ構造を考える. ここでは, 線形のリスト構造を用いる[5].

まず, 行列 A は行方向の線形リスト (以下, 行リストと呼ぶ) で持つ. 一例を Fig. 5.1 に示す. 行リストの各要素は3つの部分からできている. 一段目が行列の要素の値, 二段目が列のインデックス, 三段目がその行の次の非零要素の場所を示すポインタである. その行リストの最初の各要素へのポインタを行 index という配列に記憶する. また, 行の交換をインデックス操作で行なうために, 行 label という配列を用いる. 行 label の各要素は行 index の各要素の場所を示している.

また, 行リストと同様に列方向の線形リスト (以下, 列リストと呼ぶ) を作る. そのために列 index, 列 label という配列を用意しておく. 列リストは行リストと同じ構造を持っている. ただし, 列リストの各要素の二段目には行のインデックスが入ることを注意しておく.

次に, ピボット選択を行なう. まずピボットのある列 (以下, ピボット列と呼ぶ) を求めて列 label の付け替えで列交換を行なう. Fig. 5.1 の場合, 二列がピボット列となるので列 label の第一要素と第二要素が入れ換る. 次にピボット列からピボットを求めて行 label の付け替えを行なう. Fig. 5.1 の場合, 行 label の第一要素と第二要素が入れ換る. そして, ピボットにあたる行列要素を列リストに複写し, 残りのピボット列の要素をポインタの差し換えにより,

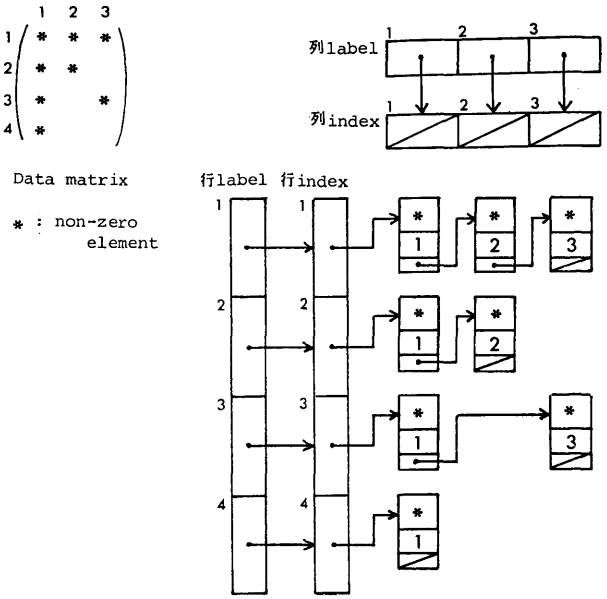

Fig. 5.1 Data structure (matrix A)

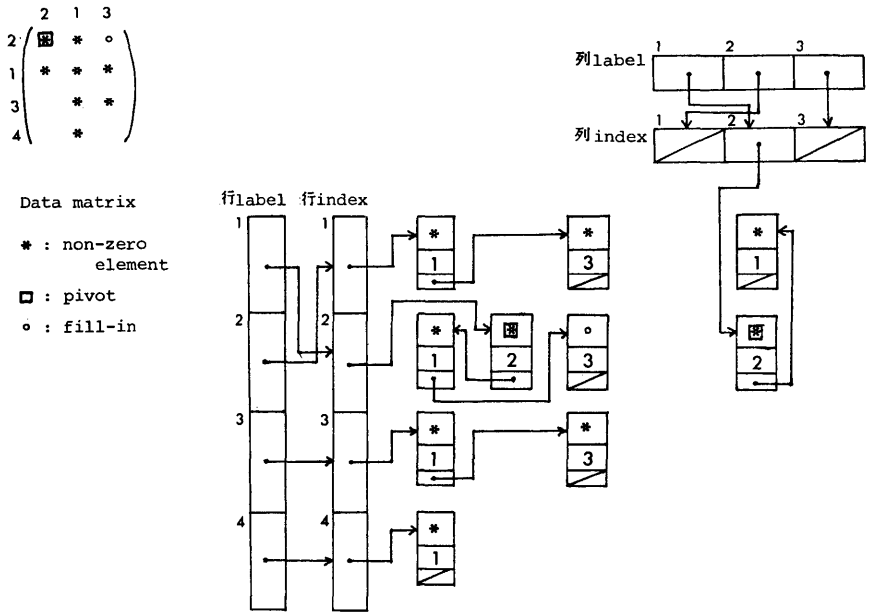

Fig. 5.2 Data strucure

Givens 法で消去される順に行リストから切り離して列リストにつなげる．このとき，要素の二段目は列のインデックスが行のインデックスに変わる．これが Q の情報となる．この Q の情報を用いて Givens 変換を行ない，fill-in の起こった場所の要素を行リストの中に挿入する．例の場合，新しい a_{13} 要素が fill-in で新しく挿入されている (Fig. 5.2)．

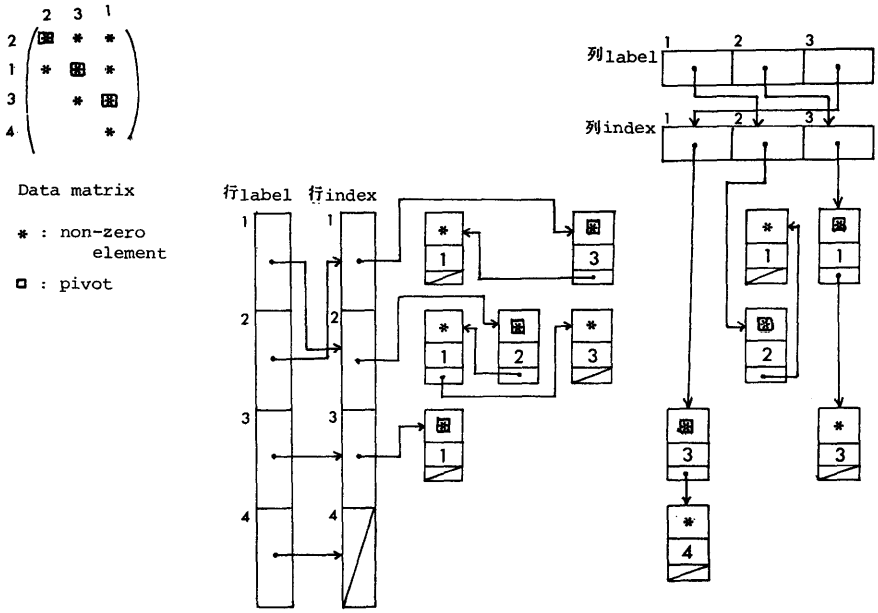

Fig. 5.3 Data structure (matrix Q and R)

ピボットにあたる行列要素は行リスト，列リスト両方に，それぞれ行 index，列 index から指定された先頭要素としてたくわえられる．

この操作を繰り返し行なうことによって，最終形は，列リストに Q の情報が，そして行リストに R がたくわえられる (Fig. 5.3)．

(one-off) の場合は Q の情報を保つ必要がないので，列ごとに Givens 変換が終わったら列 index からのポインタを切ってしまう．よって，最終形は R のみになり列 index は最初の状態になる．

6. シミュレーション

Givens 法のアルゴリズムを評価するために Householder 法を比較の対象としてシミュレーションを行なった．ここでは，モデルの行列の非零要素の配置とその要素の値を一様乱数を用いて作成した．

6.1

Fig. 6.1 に示したのは，スパースな行列のモデルとそれを QR 分解したあとの行列の非零要素のようすである．印のある場所の要素が非零要素である．

行列の大きさは (100×50) で非零要素数の最初の密度は 5% である．2〜5 番目までは各解法による QR 分解後の形である．行列を (a_{ij}) とすると，$i \leq j$ の部分が R であり，$i > j$ の部分が Q の情報を表現している部分である．(Q の情報の $i = j$ の要素は省略．)

各行列の下の数字は，(many-off) が Q の情報と R の部分の非零要素数の和の密度であり，(one-off) は Q の情報を保持しない場合の最大時の非零要素の密度である (6.2 参照)．

図からわかるように Householder 法よりも Givens 法の方が fill-in の起こり方が比較的少ない．前者は一つ非零要素が現れるとそのあとのその行の要素はほとんど確実に非零要素にな

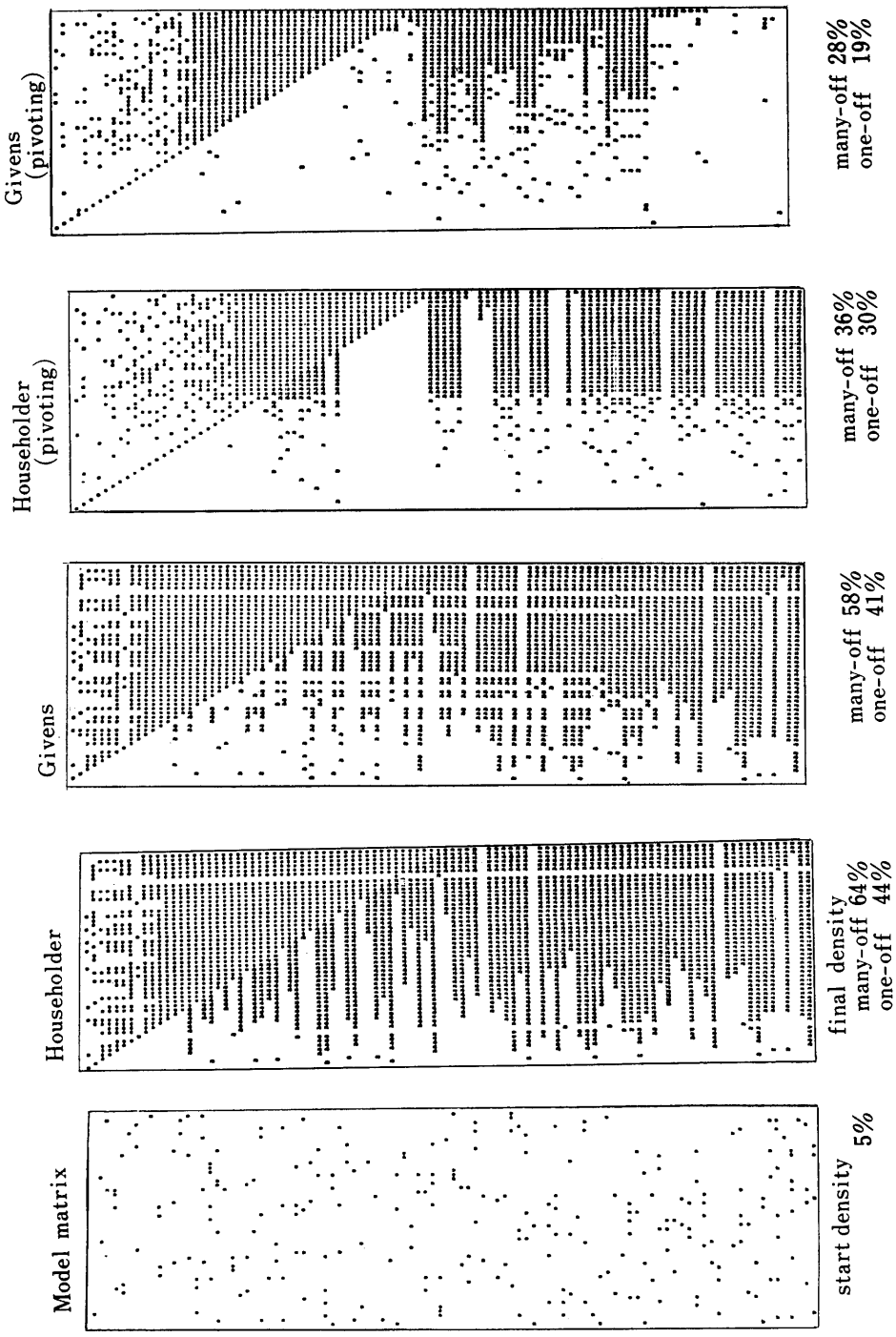

Fig. 6.1 The state showing that non-zero elements are increasing on the decomposition phase

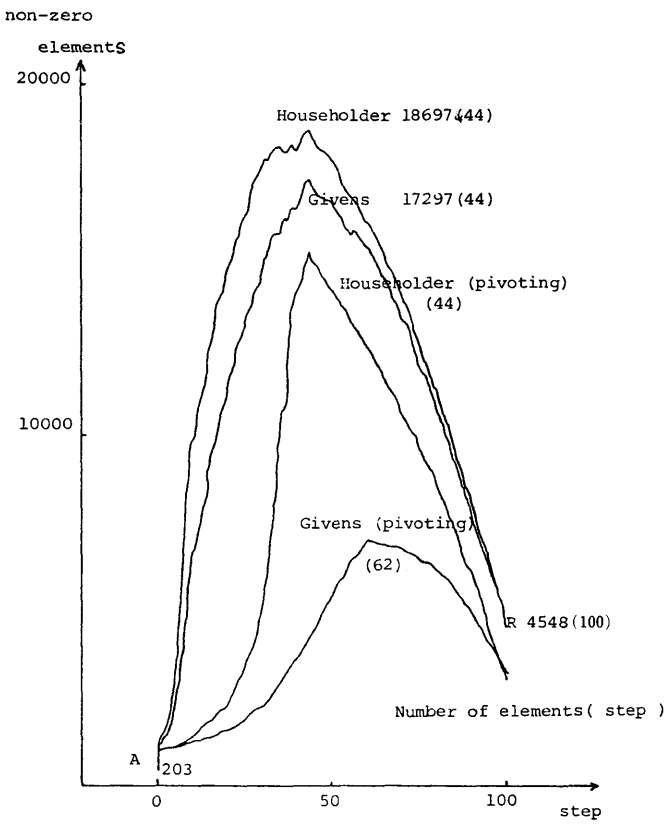

Fig. 6.2 Number of non-zero elements, data matrix (500×100)

ってしまう．それは，Givens 変換が一つ一つの非零要素に対しての変換を行なうのに対して，Householder 法は1列ずつまとめて変換を行なうためである．

また，ピボット選択によって，それぞれ半分近くに非零要素数が減ることもわかる．

6.2

Fig. 6.2 に Q の情報を保持しない場合の (one-off) のステップごとの非零要素数の変化を示す．データのモデル行列は (500×100) で非零要素数は 203 である．Q は各ステップでいらなくなった情報をすててしまう．したがって，ある程度までは fill-in によって非零要素はふえるが，ステップが進むと，fill-in の量よりもすてられる Q の情報の方が多いために全体の非零要素数はへり，最終的には R の部分の非零要素のみが残ることになる．このとき必要な記憶容量はグラフの頂点の時である．

非零要素数が最大になるまでのようすをみると，ピボット選択を行なうことによってグラフが下に凸になることがわかる．このことは，スパース性の保持が強いことを示し，大幅に計算量をへらすことになる (Fig. 6.11)．

6.3

データ行列の大きさを変えて fill-in のようすをみる．

シミュレーションは，パラメタの数 n を変えずに方程式の数 m をふやすようにした．最初の非零要素数は密度 2.5% とほぼ一定にした．そして，各大きさについて10回測定し，QR

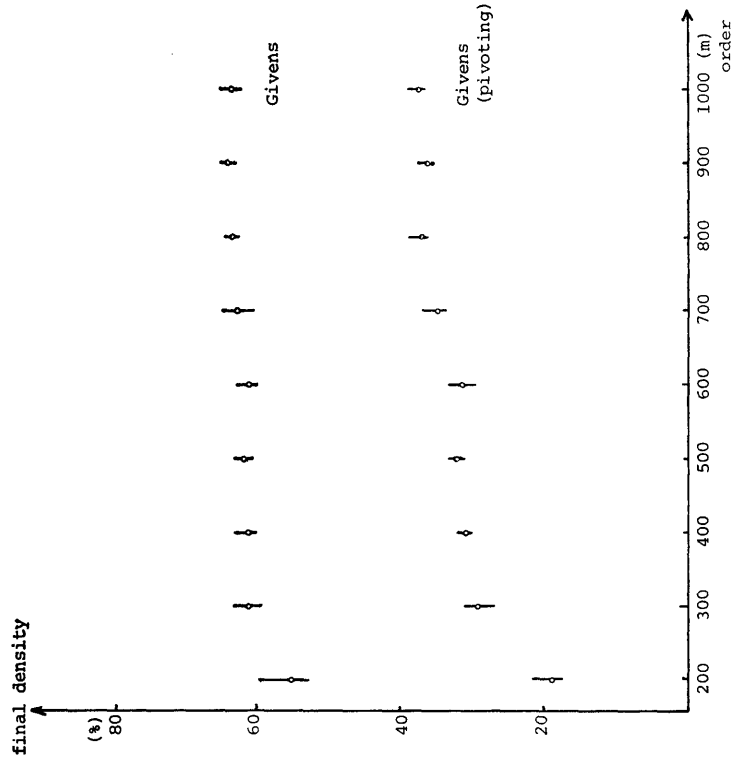

Fig. 6.4 Behavior of fill-in (Givens), keep holding on Q, data matrices ($m \times 100$), start density 2.5%

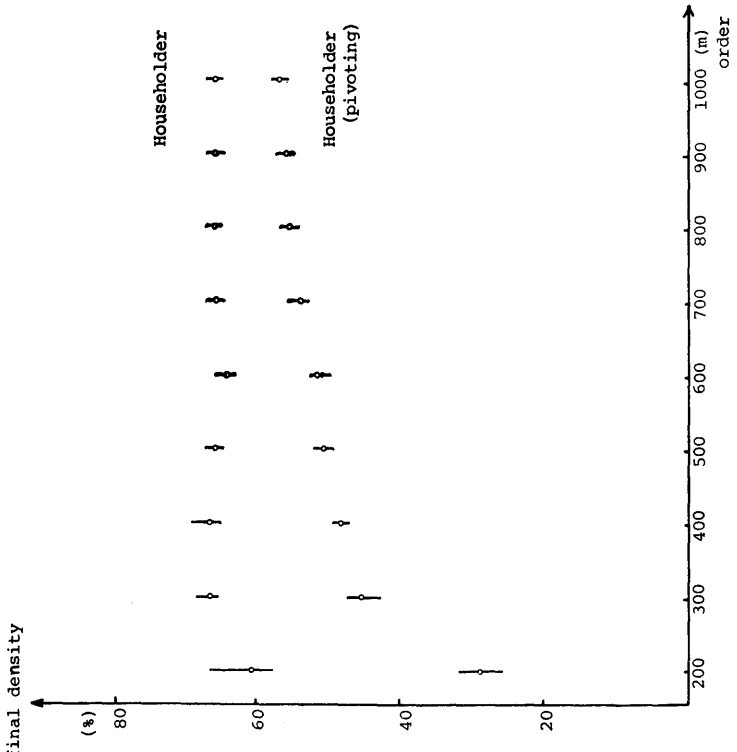

Fig. 6.3 Behavior of fill-in (Householder), keep holding on Q, data matrices ($m \times 100$), start density 2.5%

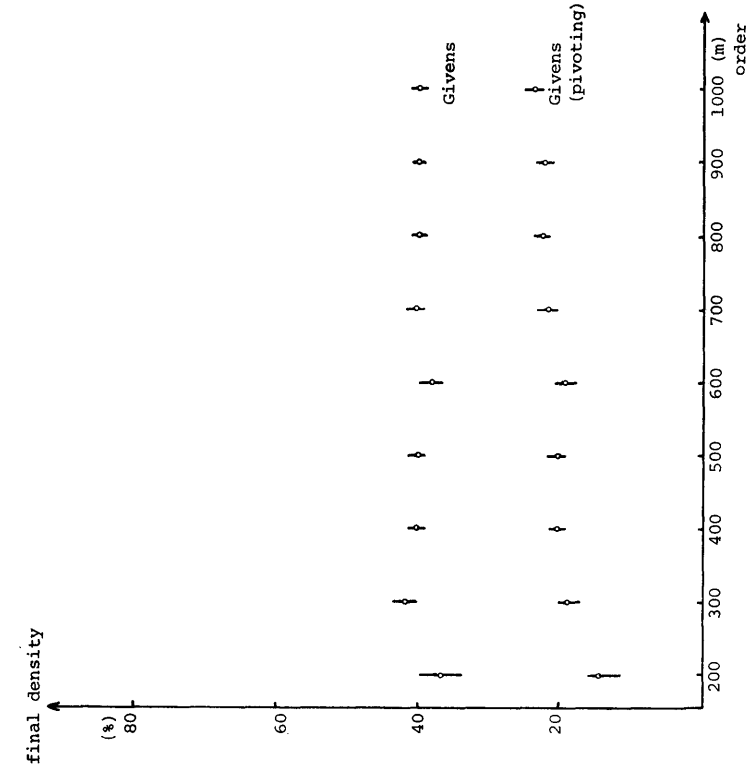

Fig. 6.6 Behavior of fill-in (Givens), data matrices ($m \times 100$), start density 2.5%

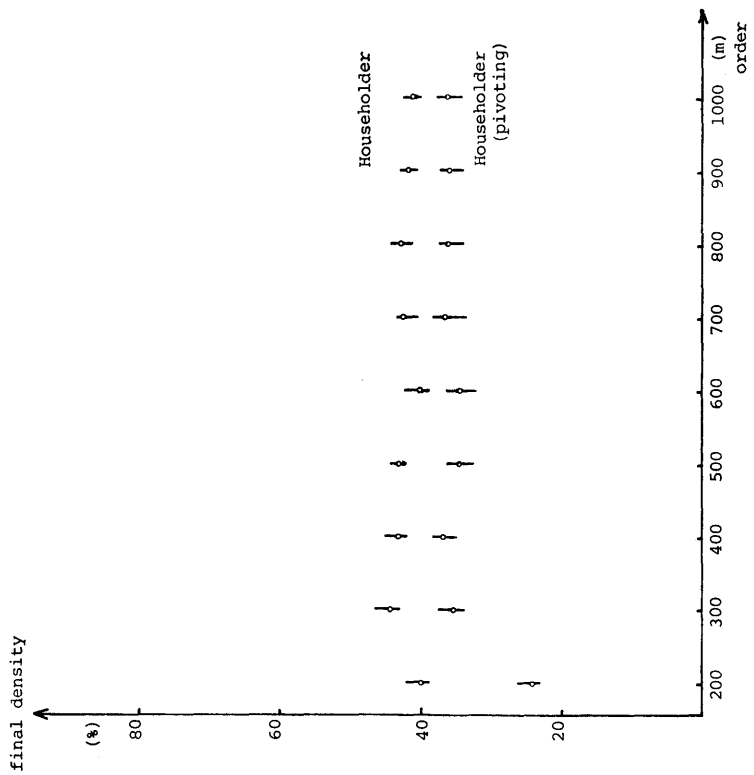

Fig. 6.5 Behavior of fill-in (Householder), data matrices ($m \times 100$), start density 2.5%

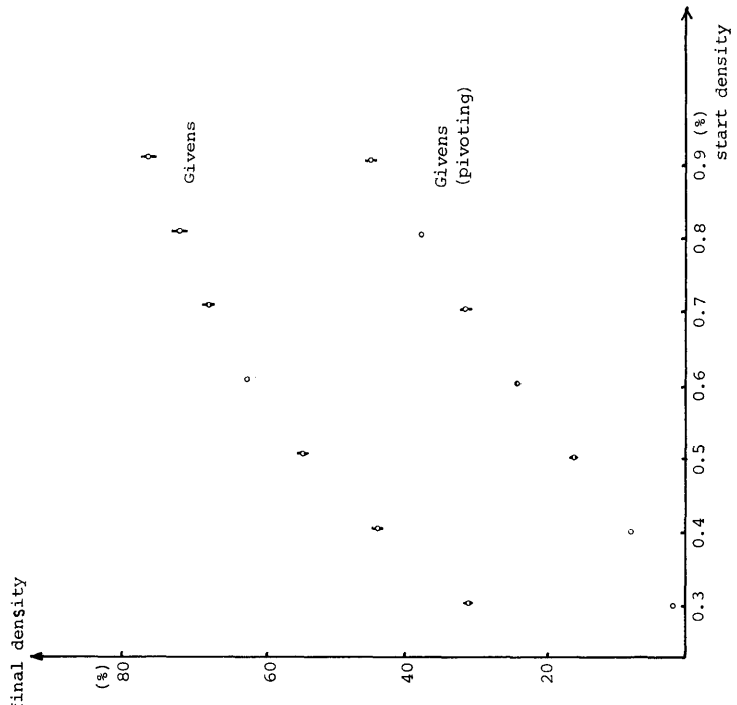

Fig. 6.8 Behavior of fill-in (Givens), keep holding on Q, data matrices (1000×500)

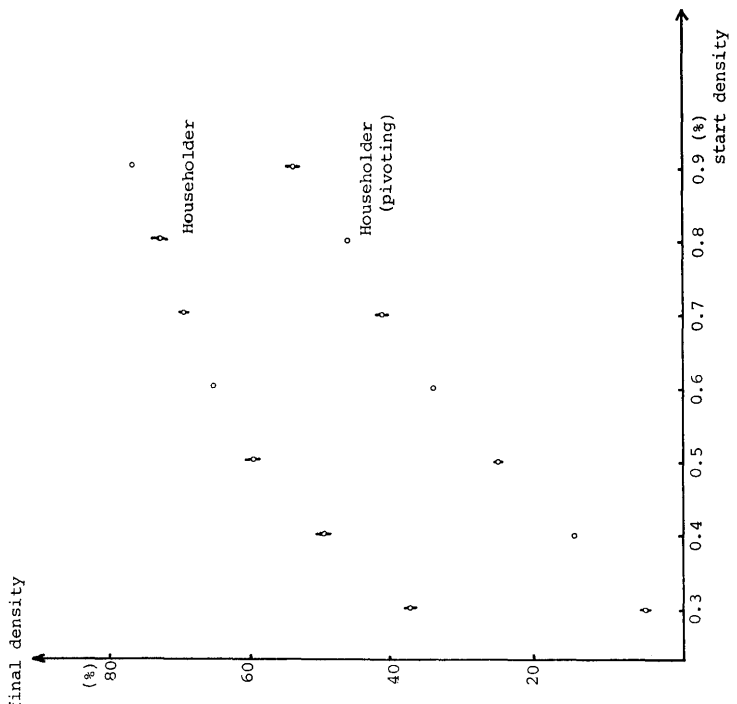

Fig. 6.7 Behavior of fill-in (Householder), keep holding on Q, data matrices (1000×500)

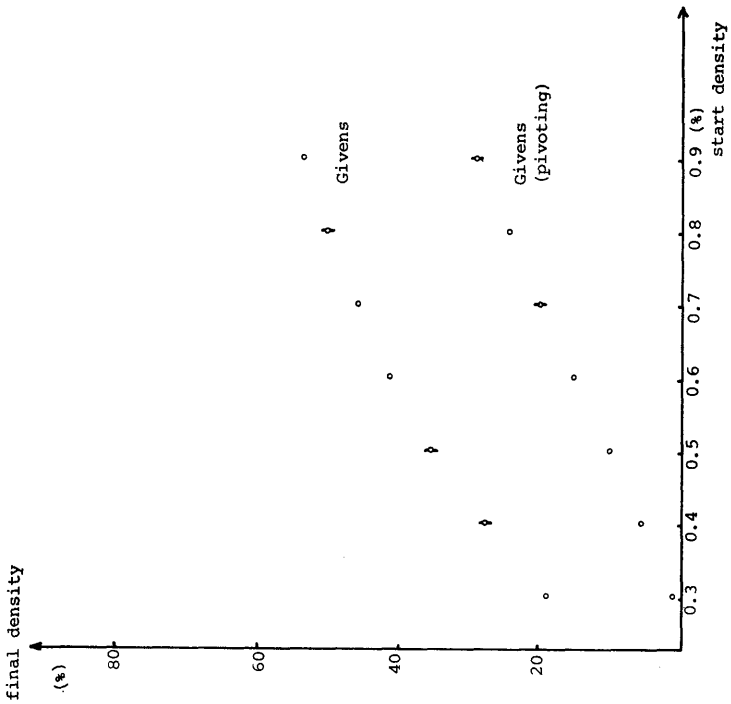

Fig. 6.10 Behavior of fill-in (Givens), data matrices (1000×500)

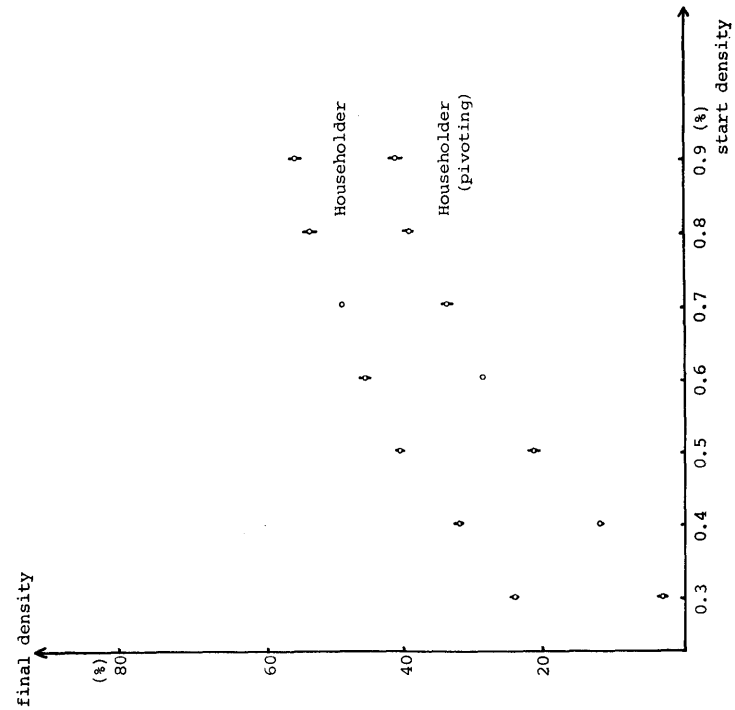

Fig. 6.9 Behavior of fill-in (Householder), data matrices (1000×500)

分解後の非零要素数の平均，最大値，最小値を示した．Fig. 6.3～Fig. 6.6 にその結果を示す．方程式の数 m をふやしてパラメタの数 n との比を大きくすることによって，最終の非零要素の密度がだんだんと一定の値に落ちついていくことがわかる．これから計算に必要な記憶量がどの程度になるかの目安となる．

次に，モデルの行列の大きさを一定にして非零要素の密度を変化させてみる．その結果は Fig. 6.7～Fig. 6.10 に示す．

6.4

今までは記憶容量の面から見てきたが，最後に計算量についてみる．その結果を Fig. 6.11 に示す．ピボット選択を行なわない場合 Givens 法の方が Householder 法よりもはるかに計算量が多い．ピボット選択を行なう場合には，fill-in の減り方が Givens 法の方が大きいために Householder 法の方が計算量は多くなる．ここで計測したのは乗除算の数である．また実線は密行列に対しての理論的な計算量である．

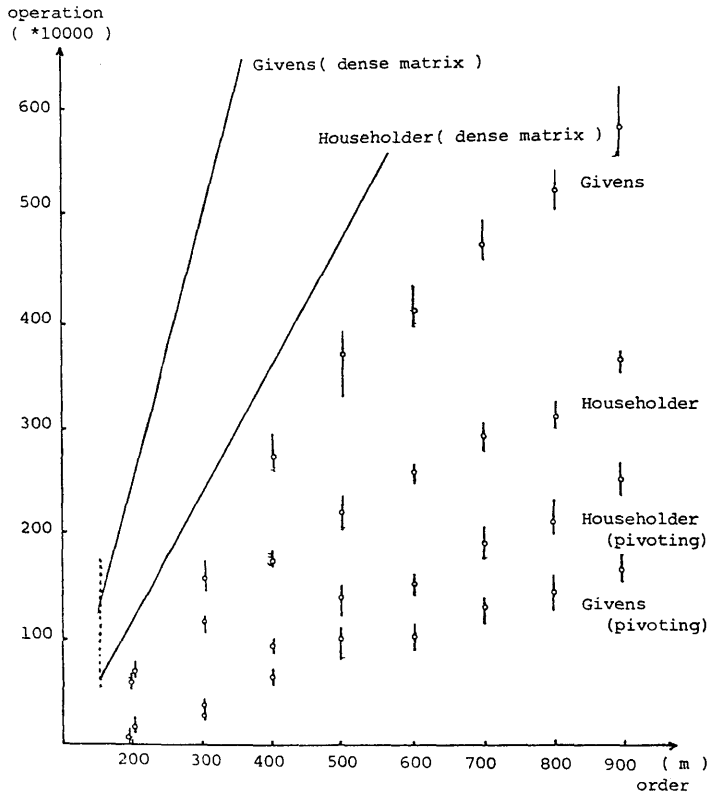

Fig. 6.11 Operation count (Decomposition phase), data matrix ($m \times 100$), start density 2%

Givens 法は，Householder 法に比べてはるかに平方根計算が多い (Householder 法で n 回，Givens 法では消去する非零要素の数)．Givens 法も Householder 法なみに平方根の計算をへらすことができる [4]．しかし，今回はピボット選択を fill-in をへらすことに重点をおいたためにそのアルゴリズムは用いなかった．しかし，実際問題上，行列が大型になれば全計算量

内の平方根計算の割合がさがって，まったく問題にならなくなる．

7. ま と め

Givens 法，Householder 法は，幾何学的には，それぞれ回転であり鏡像である．よって，一つの非零要素を零にするアルゴリズムは符号を除いて同じになる．Householder 法は一列ずつまとめて（密行列に対しては効率よく）変換する解法であるが，Givens 法は非零要素を一つ一つ零に変換していく解法である．つまり，Givens 法と Householder 法のスパースな行列に対する効率の違いは回転と鏡像によるものではなく，その操作を要素一つ一つに対して行なうか，一列ずつまとめて行なうかということである．

シミュレーションの結果をみると，まず，記憶容量の面では Givens 法は Householder 法に比べてスパースな行列に対する解法として有効なことがわかった．しかし，この解法を実用化するには今回の方法のままでは問題がある．今回は解法の性質上，データ構造にダイナミックな構造として線形リストを用いたわけであるが，その要素は行列の値，インデックス，ポインタと2つも余分な情報を持たなければならない．よって，最大値の密度がある程度小さくなければ意味がない．また，ページングなどの理由により仮想記憶マシンには不向きな構造である．これらの点を改良しなければならないだろう．

計算量の面では，初め記憶容量をへらすことに解法の重点をおいてきたにもかかわらず，Fig. 6.11 の結果から計算量の面でもスパースな問題に対しては Givens 法の方がよいことがわかった．

ピボット選択において，これはアルゴリズムからわかるように，ソーティングを含んでいる．よって要素が多くなると非常に計算量がふえる．そこで，適当なところでソーティングを打ち切ることを考える．たとえば Fig. 6.2 で頂点をすぎてから，2, 3 ステップ後にピボット選択をやめても，ふえる非零要素の数は少しであることがわかった．この点をくわしく調べることにより，実用的なものになるだろう．

実用上の行列は非零要素数が行列の大きさに比例して増えず，行列が大きくなればなるほどスパース度が強くなってくるものが多い．また，そうなるほど Givens 法の有効性は今回のシミュレーションより大きくなる．

今回はモデルの構造を考えずに，乱数で作ったデータのみのシミュレーション結果を示したが，実用上の行列は構造を持つものが多い．その構造を利用して，より効率のよい Givens 法を用いたアルゴリズムを考えることができる．この問題については稿を改めて論じる [10].

謝 辞

この研究は，統計数理研究所 田辺國士博士の指導のもとに行なったものです．また，同研究所 岸野洋久研究員，中村 隆研究員，レフリーの方々にはいろいろと有意義な助言を頂きました．心から感謝いたします．

参 考 文 献

[1] Duff, I.S. (1974). Pivot selection and row ordering in Givens reduction on sparse matrices, *Computing*, **13**, 239–248.

[2] Duff, I.S. and Reid, J.K. (1976). A comparison of some methods for the solution of sparse overdetermined systems of linear equations, *J. Inst. Math. Appl.*, **17**, 267–280.

[3] Forsythe, G.E. and Moler, C.B. (1967). Computer Solution of Linear r Algebratic System, Prentice-Hall. (渋谷・田辺訳 (1969). 線形計算の基礎, 培風館.)

[4] Gentleman, W.M. (1973). Least squares computations by Givens transformations with-

out square roots, *J. Inst. Math. Appl.*, **12**, 329–336.
[5] Knuth, D.F. (1973). *The Art of Computer Programming*, vol. 1, *Fundamental Algorithms*, second edition.
[6] 名取 亮 (1972). 大型行列の固有値問題, 情報処理, **13**, 29-36.
[7] 小柳義夫 (1982). 最小二乗法の新しいアルゴリズム, 情報処理, **23**, 99-108.
[8] Reid, J.K. (1976). Solution of Linear Systems of Equations: Direct Methods (general), A. Dold and B. Eckmann, **572** Sparse Matrix Techniques, Lecture Notes in Mathematics, Springer Verlag.
[9] 渋谷政昭 (1971). 最小二乗法のアルゴリズム, 応用統計学, **1**, 47-60.
[10] 田中輝雄, 田辺國士 (1983). ベイズの方法によるデータのあてはめ, 京大数理解析研講究録, 発表予定.

Givens' Method and Householder's Method for Solution of Sparse Least Squares Probrem

Teruo Tanaka
(University of Electro-communications
and The Institute of Statistical Mathematics)

Givens' method and Householder's method for the least squares solution of sparse systems of large linear equations are compared from the point of view of storage. It is shown experimentaly in this paper that Givens' method has some advantage in storage and operation counts over Householder's method in case of sparse matrices while the latter is more efficient in general.