アルゴリズム 16－1
（1990．7．16）

大型疎線形計画問題に対するReidの基底更新アルゴリズムの改善とその性質

大柳俊夫　　大内　東
北海道大学工学部

大型の線形計画問題の基底行列は一般に疎であり，計算機の記憶容量や処理能力などの制約のもとで正確かつ高速に問題を解くためには，基底更新の際にその疎な性質をいかに保持するかが問題となる．この問題に対して，R.H.Bartels と G.H.Golub は，基底行列を三角分解する方法を提案し，その後 J.K.Reid は，その方法に改良を加え，基底更新の際に行および列の置換を行ってバンプを縮小し，フィル・インを抑える効率の良い方法を提案した．

本論文では，Reid 法のバンプ縮小の基本的性質を明らかし，その性質を利用してバンプ縮小の効率改善を図った改良 Reid 法を提案する．また，改良 Reid 法と Reid 法のバンプ縮小効率の関係を明らかにする．

On Improvement of Reid's Basis Updating Method for Large Sparse Linear Programming Problems and Its Characteristic.

Toshio Ohyanagi Azume Ohuchi
Faculty of Engineering,
Hokkaido University
North 13 West 8, North ward, Sapporo 060, Japan

This paper intends to investigate some inherent characteristics of Reid's basis updating method (Reid Method) for linear programming problems and to improve an efficiency of the method. Reid method is explained in detail and then seven lemmas concerned in its characteristics are clarified.An improved basis updating method (improved Reid method) is developed by making good use of its characteristics. A theorem which indicates difference of the efficiency between Reid method and improved Reid method is proved.

1．はじめに

1947年にG.B.Danzigが線形計画問題として定式化されたアメリカ空軍のある計画問題を解くために単体法を考案して以来，さまざまな産業分野における生産計画や運用計画などの問題が線形計画問題として定式化されるようになり，単体法は，計画立案・意志決定の道具として欠かせないものとなった．近年，そのような産業分野の組織の巨大化，複雑化に伴い，定式化される線形計画問題の規模も大型化してきており，大型の線形計画問題を解く必要性が高まってきている．

大型の線形計画問題の基底行列は一般に疎な構造をしており，計算機の記憶容量や処理能力などの制約のもとで正確かつ高速に問題を解くためには，基底更新の際にその疎な性質をいかに保持するかが重要な問題となる．この問題に対して，1969年に R.H.BartelsとG.H.Golub は，基底行列を三角分解して保存しておく方法を提案した[1]．その方法は，基底の逆行列を積形式にして保存する方法と比較すると，元問題の疎な性質を十分生かすことが可能で，計算時間や記憶容量の点で優れたものである． R.H.BartelsとG.H.Golub の発表以来，多くの研究者によりこの方法の改良が行われてきた[2)-6)]． その中で J.K.Reid による改良は，データがメモリー内におさまり，計算をメモリー上で行うことが可能な場合に，計算時間と計算精度の両方の点で優れたものとして知られている．Reidの方法（以下，Reid法と略す）は，基底行列の疎な性質をできる限り保持するために，基底更新の際に行および列の置換操作を行って基底行列中のバンプと呼ばれる正方部分行列を縮小し，フィル・インの直接的な原因となる消去をなるべく少なくする効率の良い方法である．

本論文では，Reid法における行および列の置換操作によるバンプ縮小の基本的性質を明らかにし，その性質を利用してバンプ縮小の効率改善を図った改良Reid法を提案する．そして，改良Reid法とReid法の間のバンプ縮小効率の関係を明らかにする．またReidの用いた例題により両方法を比較し，改良Reid法の有効性を示す．

以下，２章で基底行列の三角分解とReid法について説明し，３章でバンプ縮小の基本的性質について述べる．そして４章でその基本的性質を利用した改良Reid法を提案し，２つの方法の間のバンプ縮小効率の関係を明らかにする．

2．Ｒｅｉｄ法

2．1　大型疎線形計画問題

一般に，線形計画問題は，

目的関数：　$z = \mathbf{c}^T\mathbf{x} \rightarrow$ 最大化，　(1)

制約条件：　$\mathbf{A}\mathbf{x} = \mathbf{b}$，$\mathbf{x} \geqq 0$，　(2)

と定式化される．ここで，$\mathbf{c}$，$\mathbf{x}$はｎ次元列ベクトル，$\mathbf{b}$はｍ次元列ベクトル，$\mathbf{A}$はｍ×ｎ行列であり，

$\mathrm{rank}(\mathbf{A}) = m$，$m \leqq n$，　(3)

を仮定する．また，T は転置を表すものとする．現実の線形計画問題は，ｍやｎが1000を越えるような大規模なものであり，このような場合，一般に$\mathbf{A}$は非０成分の密度が１％以下の疎行列となる．そのため，$\mathbf{A}$の中から独立なｍ本の列ベクトルを取り出してつくられる基底行列$\mathbf{B}$も疎な構造となる．この$\mathbf{B}$に対して単体法の各反復では，新しく基底に入る変数と基底から追い出される変数を，

$\mathbf{y}\mathbf{B} = \mathbf{c}_B$,　(4)

$\mathbf{B}\mathbf{d} = \mathbf{a}$,　(5)

の２組の連立一次方程式を解いて決定し，基底を更新する．なお，記号は文献5)による．この更新は，最適解が得られるか，または非有界であることが明らかになるまで繰返される．

2．2　基底行列の三角分解とＲｅｉｄ法

Reid法では， BartelsとGolub の方法と同様に基底行列$\mathbf{B}$を三角分解して保存する．一般に，$\mathbf{B}$の行および列置換を伴った三角分解は，

$\mathbf{M}_r\mathbf{M}_{r-1}\cdots\mathbf{M}_1\mathbf{B} = \mathbf{P}\mathbf{U}\mathbf{Q}$,　(6)

と表すことができる．ただし，$\mathbf{M}_i$ は対角要素がすべて１で１つの非対角要素のみ非０である行列（単位行列と１つの非対角要素のみ異なる行列），

$\mathbf{P}$と$\mathbf{Q}$は置換行列，そして$\mathbf{U}$は上三角行列である．次の単対法の反復で基底行列$\mathbf{B}$が$\bar{\mathbf{B}}$へ更新されると(6)式は，

$$\mathbf{M}_r\mathbf{M}_{r-1}\cdots\mathbf{M}_1\bar{\mathbf{B}}=\mathbf{P}\,\mathbf{S}\,\mathbf{Q}, \qquad (7)$$

となる．したがって，行列$\mathbf{U}$と$\mathbf{S}$は，$\mathbf{B}$から$\bar{\mathbf{B}}$への更新の際に入替えが起こった1つの列に対応する列のみ異なることになる．図1に$\mathbf{S}$の構造を示す．行列$\mathbf{S}$で対角より下に非0要素のある列をスパイク列，スパイク列で対角以下の部分をスパイクと呼ぶ．図1に示すように，スパイクは第s行から第t行までのびているとする．また，スパイクを左端の列として含む最小の正方部分行列つまり第s行から第t行と第s列から第t列で囲まれた部分行列をバンプと呼ぶ．バンプ内に非0要素が1つだけ存在する列のその非0要素自身を列シングルトンと呼ぶ．同様に，行に関して行シングルトンを定義する．

さて，行列$\mathbf{S}$を三角分解して$\mathbf{U}$を$\bar{\mathbf{U}}$へ更新した結果は，

$$\mathbf{M}_{\bar{r}}\mathbf{M}_{\bar{r}-1}\cdots\mathbf{M}_{\bar{1}}\mathbf{M}_r\cdots\mathbf{M}_1\bar{\mathbf{B}}=\bar{\mathbf{P}}\,\bar{\mathbf{U}}\,\bar{\mathbf{Q}}, \qquad (8)$$

と表すことができる．この更新におけるフィル・インの発生を抑制するには，その直接的な原因となる消去操作，つまり(8)式における$\mathbf{M}_{\bar{1}},\cdots,\mathbf{M}_{\bar{r}}$行列の積演算をできる限り少なくすることが重要である．そのためには，消去操作を行う前にバンプをできる限り縮小する行および列の置換が必要となる．Reid法はそのための効率的な方法の1つである．図2にReid法のアルゴリズムを示す．

Reid法では，まず列シングルトンのバンプ左上隅への移動によりバンプを縮小する（図2の第1行～第11行）．具体的には，列シングルトンを第s＋1列から順に探索し，列シングルトンが第k列に存在した場合，行列$\mathbf{S}$の列と行に対して，

$$\begin{pmatrix}1 & \cdots & s\text{-}1 & s & s\text{+}1 & \cdots & k\text{-}1 & k & k\text{+}1 & \cdots & t & \cdots & m\\ 1 & \cdots & s\text{-}1 & s\text{+}1 & s\text{+}2 & \cdots & k & s & k\text{+}1 & \cdots & t & \cdots & m\end{pmatrix} \qquad (9)$$

という置換操作を行う．つまり列に対して，第s列から第k－1列までをそれぞれ第s＋1列から第k列へ移し，同時に第k列を第s列へ移す．その後，行に対しても同様の移動を行う．この置換により$\mathbf{S}$内のスパイク列の位置は更新されバンプは縮小される．この操作は列シングルトンが存在する間続けられる．

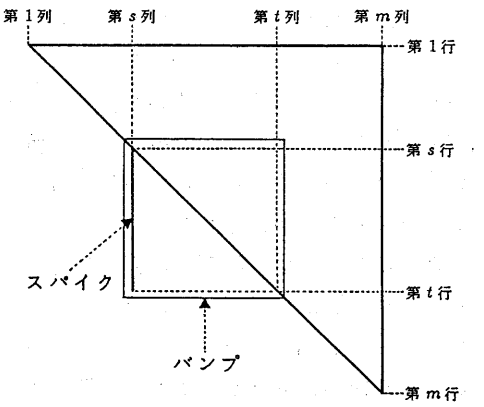

図1　行列Sのスパイクおよびバンプ．
Fig.1　Spiked matrix S and its bump.

```
   (* 列シングルトンの移動によるバンプの縮小 *)
 1: repeat
 2:     k := s + 1;
 3:     find_singleton := false;
 4:     while (k <= t) and (not find_singleton) do
 5:         if < 第 k 列に列シングルトンが存在する > then
 6:             find_singleton := true
 7:         else
 8:             k := k + 1;
 9:     if ( find_singleton ) then
10:         < 第 k 列の列シングルトンをバンプ左上隅に移動する >
11: until ( not find_singleton ) or < バンプが消滅する >;
12: if < バンプが存在する > then begin
       (* 行シングルトンの移動によるバンプの縮小 *)
13:     repeat
14:         k := t - 1;
15:         find_singleton := false;
16:         while (k >= s) and (not find_singleton) do
17:             if < 第 k 行に行シングルトンが存在する > then
18:                 find_singleton := true
19:             else
20:                 k := k - 1;
21:         if ( find_singleton ) then
22:             < 行シングルトンをバンプ右下隅に移動する >
23:     until ( not find_singleton );
       (* 第 t 列の列シングルトンの移動によるバンプの縮小 *)
24:     < 列の置換によりバンプを upper Hessenberg 形式にする >;
25:     while < 第 t 列に列シングルトンが存在する > do
26:         < 第 t 列の列シングルトンをバンプ左上隅に移動する >;
       (* 副対角要素の消去による三角分解 *)
27:     if < バンプが存在する > then < 副対角要素を消去する >
28: end ;
```

図2　Reid法のアルゴリズム．
Fig.2　An algorithm of Reid method.

次に行シングルトンのバンプ右下隅への移動によりバンプを縮小する（図2の第13行～第23行）．具体的には，第t－1行から逆順に行シングルトンを探索し，第k行に行シングルトンが存在した場合，行列$\mathbf{S}$の行と列に対して，

$$\begin{pmatrix}1 & \cdots & s & \cdots & k\text{-}1 & k & k\text{+}1 & \cdots & t\text{-}1 & t & t\text{+}1 & \cdots & m\\ 1 & \cdots & s & \cdots & k\text{-}1 & t & k & \cdots & t\text{-}2 & t\text{-}1 & t\text{+}1 & \cdots & m\end{pmatrix} \qquad (10)$$

という置換操作を行う．つまり行に対して，第k＋1行から第t行までをそれぞれ第k行から第t－1行へ移し，同時に第k行を第t行へ移す．その後列に対しても同様の移動を行う．この置換操作によりS内のバンプの最後の行の位置は更新されバンプは縮小される．この操作は行シングルトンが存在する間続けられる．

さらにその後で，行列Sの列に対して，

$$\begin{pmatrix} 1 & \cdots & s-1 & s & s+1 & \cdots & t-1 & t & t+1 & \cdots & m \\ 1 & \cdots & s-1 & t & s & \cdots & t-2 & t-1 & t+1 & \cdots & m \end{pmatrix} \quad (11)$$

という置換操作を行う．つまり，第s＋1列から第t列までをそれぞれ第s列から第t－1列へ移し，同時に第s列を第t列へ移す．この置換でバンプはupper Hessenberg形式となり（図3），バンプの最後の列に列シングルトンが存在する可能性が出てくる．その場合は，バンプの最後の列に列シングルトンが存在する間，列シングルトンのバンプ左上隅への移動を繰返し行いバンプを縮小する（図2の第24行～第26行）．

以上のバンプ縮小操作の途中で，列シングルトンの移動によりバンプが消滅することが起こり得る．その場合は，三角分解が完了したことになりアルゴリズムを終了する．まだバンプが存在している場合は，副対角要素の消去により三角分解を完了する（図2の第27行）．

以上で述べたReid法のアルゴリズムをまとめると，まず列シングルトンの移動によりバンプを縮小し，次に行シングルトンの移動によりバンプを縮小する．その後，バンプをupper Hessenberg形式にし，再度，列シングルトンの移動によりバンプを縮小する．そして最後に，副対角要素を消去して三角分解を完了する．

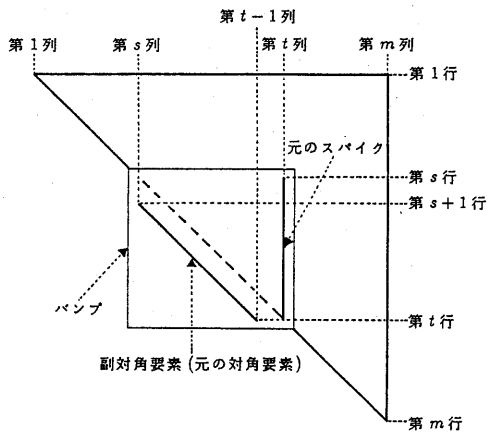

図 3　upper Hessenberg 行列.
Fig.3　upper Hessenberg matrix.

3．バンプ縮小の基本的性質

Reid法の行および列の置換操作によるバンプ縮小について詳細に検討した結果，以下の補題で示される基本的性質が明らかとなった．なお，補題1から6はバンプをupper Hessenberg形式にする前の性質，補題7はその後の性質である．また，補題2と3については文献4)で証明はされていないが簡単に述べられていることである．

補題1　列シングルトンが第s列と第t列に同時に存在することはない．

証明　第s列と第t列に列シングルトンが同時に存在していると仮定する．すると，第t列は第1列から第s列までのs個の列ベクトルの線形結合で表すことが可能となる．したがってSの階数は，

$$\mathrm{rank}(\mathbf{S}) \leq m-1, \quad (12)$$

となる．また，初期基底行列が正則の場合，単体法ではその後の基底行列は正則に保たれることから，すべての反復において，

$$\mathrm{rank}(\bar{\mathbf{B}}) = m, \quad (13)$$

が成立する．一方，$\mathbf{M}_i$が正則であることから(7)式は，

$$\bar{\mathbf{B}} = \mathbf{M}_1^{-1} \cdots \mathbf{M}_r^{-1} \mathbf{P}\,\mathbf{S}\,\mathbf{Q}, \quad (14)$$

となり，(12)式より右辺の階数は，

$$\mathrm{rank}(\mathbf{M}_1^{-1} \cdots \mathbf{M}_r^{-1} \mathbf{P}\,\mathbf{S}\,\mathbf{Q}) \leq m-1, \quad (15)$$

となる．これは，左辺の階数がmであることと矛盾する．ゆえに，列シングルトンが第s列と第t列に同時に存在することはない（証明終）．

補題2　列シングルトンの存在する列が第s列および第t列以外の場合は，列シングルトンの移動によりバンプの次数は正確に1減少される．

証明　列シングルトンの移動前のバンプの次数

D_1 は，スパイクが第 s 行から第 t 行までのびていることから，

$$D_1 = t - s + 1, \quad (16)$$

である．一方，列シングルトンの移動は(9)式の置換で表されることから，列シングルトン移動後のスパイクは第 s + 1 行から第 t 行までのびていることになる．したがって列シングルトン移動後のバンプの次数 D_2 は，

$$D_2 = t - (s+1) + 1 = t - s, \quad (17)$$

となる．ゆえに，バンプの次数は正確に 1 減少される（証明終）．

補題3　行シングルトンの移動によりバンプの次数は正確に 1 減少される．

証明　行シングルトンの移動は(10)式の置換で表されることから，行シングルトン移動後のスパイクは，第 s 行から第 t − 1 行までのびていることになる．したがって行シングルトン移動後のバンプの次数 D_3 は，

$$D_3 = (t-1) - s + 1 = t - s, \quad (18)$$

となる．一方，行シングルトン移動前のバンプの次数は D_1 である．ゆえに，バンプの次数は正確に 1 減少される（証明終）．

補題4　第 t 列に列シングルトンが存在する場合，それを移動することによりバンプの次数は少なくとも 1 減少される．最良の場合，その移動により三角分解が完了する．

証明　列シングルトンが第 t 列の場合の置換は，

$$\begin{pmatrix} 1 & \cdots & s-1 & s & s+1 & \cdots & t-1 & t & t+1 & \cdots & m \\ 1 & \cdots & s-1 & s+1 & s+2 & \cdots & t & s & t+1 & \cdots & m \end{pmatrix} \quad (19)$$

となり，第 s 列から第 t − 1 列まではそれぞれ第 s + 1 列から第 t 列へ移り，同時に第 t 列は第 s 列へ移る．同様に行の移動も行われる．したがって，置換操作以前にバンプの最後の行であった第 t 行がバンプの外へ追い出されることになり，移動後のスパイクは第 t 行までのびているとは限らないことになる．スパイクがどこまでのびているかはスパイク列の非 0 要素の存在する位置に依存することになる．いまスパイクが第 s + 1 行から第 k 行($s+1 \leqq k \leqq t$)までのびているとすると，列シングルトン移動後のバンプの次数 D_4 は，

$$D_4 = k - (s+1) + 1 = k - s, \quad (20)$$

となり，k の取り得る範囲を考慮すると，

$$1 \leqq D_4 \leqq t - s, \quad (21)$$

となる．したがって列シングルトン移動前後のバンプの次数の差は，

$$1 \leqq D_1 - D_4 \leqq t - s, \quad (22)$$

となる．ゆえに，列シングルトンの移動によりバンプの次数は少なくとも 1 減少される．また $k = s+1$ の場合，(20)式より $D_4 = 1$ となり，三角分解は完了する（証明終）．

補題5　行シングルトンの移動により，バンプ内に新しく列シングルトンが現れることはない．

証明　バンプ内の各列で行シングルトンの存在する行に対応する各列の要素の値は 0 である．したがって，(10)式で表される置換操作により行シングルトンの存在する行がバンプの外へ追い出されても，移動後のバンプ内の各列の非 0 要素数は移動前と変りないことになる．したがって，行シングルトンの移動により新しく列シングルトンが現れることはない（証明終）．

補題6　行シングルトンの移動によりバンプが消滅することはない．

証明　バンプの最後の行の非 0 要素は，スパイクと対角上の 2^n 所に必ず存在し，行シングルトンの移動を何度繰返してもそれら 2 つの非 0 要素はバンプ内から消えることはない．したがって，行シングルトンの移動によりバンプが消滅することはない（証明終）．

補題7　バンプを upper Hessenberg 形式にした後の第 t 列の列シングルトンの移動によりバンプの次数は正確に 1 減少される．

証明　置換は(19)式で表されるので，バンプの最初の列は第 s 列から第 s + 1 列へ移る．一方，列シングルトン移動後も副対角要素はすべて非 0 であるから第 t 行，第 t − 1 列成分も非 0 で，移動後のバンプの次数は D_2 となる．したがって，この列シングルトンの移動によりバンプの次数は正確に 1 減少される（証明終）．

4．改良Ｒｅｉｄ法

4．1　改良Ｒｅｉｄ法とその性質

3章で示したバンプ縮小の基本的性質を利用して，

(1) 第ｔ列に列シングルトンが存在する場合，この列シングルトンの移動を優先する，

(2) 第ｓ列に列シングルトンが存在する場合，第ｓ列と第ｔ列を交換する，

ようにReid法を改良する．

(1) に関しては，補題4より，この列シングルトンをバンプ左上隅に移動することによりバンプが大きく縮小される可能性がある．バンプを大幅に縮小することができれば，その後のシングルトンの探索回数と列および行の置換回数は少なくなることが期待され，バンプ縮小の効率化を図ることができると考えられる．

(2) に関しては，補題1より，第ｓ列と第ｔ列に同時に列シングルトンが存在しないことが保証されており，第ｓ列と第ｔ列を交換することにより上で述べた(1) と同様のバンプ縮小の効率化が期待できる．Reid法では，第ｓ列は列シングルトンの探索範囲から除かれており，第ｓ列に列シングルトンが存在する場合，その列シングルトンはupper Hessenberg形式にした後で移動されることになる．そのときのバンプの縮小幅は補題7より1であり大幅な縮小とはならない．また，行シングルトン移動前に，第ｓ列と第ｔ列の交換を行ってバンプ内のすべての列シングルトンを移動すると，補題5より，バンプをupper Hessenberg形式にした後での列シングルトンの移動は不要となる．

以上のことを考慮してReid法を改良した改良Reid法のアルゴリズムを図4に示す．図中の第1行～第19行が列シングルトンの移動によりバンプを縮小する部分である．この中で第3行および第5行が今回の改良の中心部分で，それぞれ第ｔ列および第ｓ列の列シングルトンの有無を調べる部分である．また，第21行～第31行が行シングルトンの移動によりバンプを縮小する部分で，これはReid法の場合とまったく同じである．そして第32行

```
    (* 列シングルトンの移動によるバンプの縮小 *)
 1: repeat
 2:     find_singleton := true;
 3:     if < 第 t 列に列シングルトンが存在する > then
 4:         k := t
 5:     else if < 第 s 列に列シングルトンが存在する >then begin
 6:         < 第 s 列と第 t 列を交換する >
 7:         k := t
 8:     end else begin
 9:         k := s + 1;
10:        find_singleton := false;
11:        while (k < t) and (not find_singleton) do
12:            if < 第 k 列に列シングルトンが存在する > then
13:                find_singleton := true
14:            else
15:                k := k + 1
16:    end;
17:    if ( find_singleton ) then
18:        < 第 k 列の列シングルトンをバンプ左上隅に移動する >
19: until ( not find_singleton ) or < バンプが消滅する >;
20: if < バンプが存在する > then begin
        (* 行シングルトンの移動によるバンプの縮小 *)
21:    repeat
22:        k := t - 1;
23:        find_ingleton := false;
24:        while (k >= s) and ( not find_singleton) do
25:            if < 第 k 行に行シングルトンが存在する > then
26:                find_singleton := true
27:            else
28:                k := k - 1;
29:        if ( find_singleton ) then
30:            < 行シングルトンをバンプ右下隅に移動する >
31:    until ( not find_singleton );
       (* 副対角要素の消去による三角分解 *)
32:    < 列の置換によりバンプを upper Hessenberg 形式にする >;
33:    < 副対角要素を消去する >
34: end;
```

図4　改良 Reid 法のアルゴリズム．
Fig.4　An algorithm of improved Reid method.

がバンプをupper Hessenberg形式にする部分，第33行が副対角要素の消去を行う部分である．ここで，Reid法では副対角要素の消去を行う前に図2の第27行に示すようにバンプの有無を調べているが，改良Reid法では補題6よりその必要はない．

この改良Reid法のバンプ縮小の効率に関して次の定理が成立する．

定理　改良Reid法によるバンプ縮小の際の列シングルトンと行シングルトンを合せた移動回数は，Reid法による場合よりも多くなることはない．

証明　改良Reid法は，上で述べた(1)や(2)が起こった場合にバンプを大幅に縮小させるようにReid法の列シングルトンの移動順序を変更したものである．したがって，(1)と(2)が起こらない場合は，図2と図4のアルゴリズムを比べるとわかるように，列および行置換の操作はReid法による場合とまったく同じになり，シングルトン移動回数は等しくなる．一方，(1)や(2)が起こる場合，Reid法では，その列シングルトンの探索順序と補題2で述べたことから，バンプ中に存在するシング

ルトンが他の列シングルトンの移動によりバンプの外に追い出されることは起こり得ない．これに対して改良Reid法では，補題4よりバンプの大幅な縮小の可能性が有り，そのためバンプ内に存在するシングルトンが他の列シングルトンの移動によりバンプの外に追い出されることが起こり得る．したがって，改良Reid法によるシングルトンの移動回数は，Reid法による場合に比べ少なくなる可能性がある．

以上のことから，改良Reid法によるバンプ縮小の際のシングルトン移動回数はReid法による場合よりも多くなることない（証明終）．

4．2　例題

Reid法と改良Reid法のバンプ縮小の効率を比較するために，文献4)で用いられている例題を取上げる．図5にその例題の行列**S**の構造を示す．図中の×印は非0要素，数字は行および列番号，そして行列中で破線で囲まれた部分はバンプを表す．この例題に対して改良Reid法を適用した場合の行

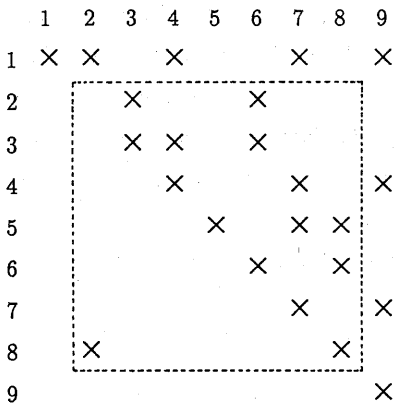

図5　例題の行列 S.
Fig.5　An example matrix S.

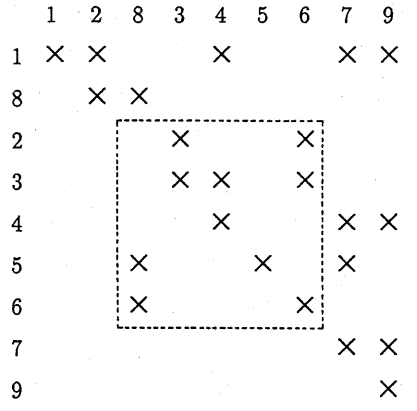

図6　最初の列シングルトン移動後の行列 S.
Fig.6　Matrix S after first column singleton moved.

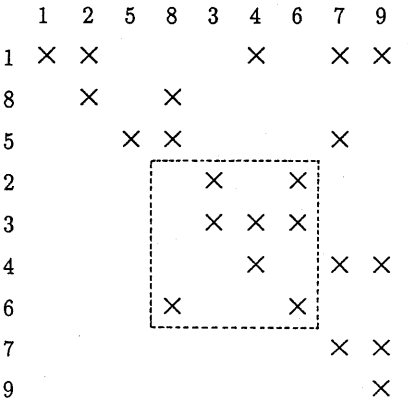

図7　2番目の列シングルトン移動後の行列 S.
Fig.7　Matrix S after second column singleton moved.

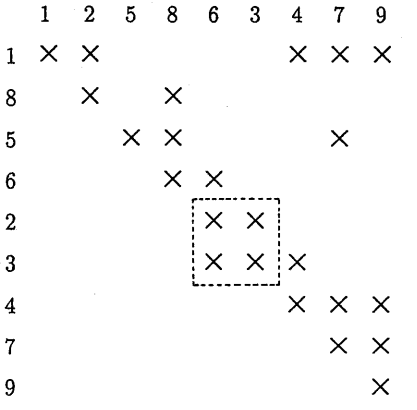

図8　最後の列シングルトン移動後の行列 S.
Fig.8　Matrix S after last column singleton moved.

列の変化を図6から図8に示す．図6は列番号2と8の列を交換後に列番号2の列の列シングルトンを移動した結果，図7は列番号5の列の列シングルトンを移動した結果，そして図8は列番号8と6の列を交換後に列番号8の列の列シングルトンを移動した結果である．このように改良Reid法では，3回の列シングルトンの移動でバンプの縮小を終える．これに対してReid法では，文献4)で示されているように，1回の列シングルトンの移動，2回の行シングルトンの移動，そしてupper Hessenberg形式にした後の2回の列シングルトンの移動の計5回のシングルトンの移動が必要である．この例題のように基底行列のサイズが小さい場合でも，Reid法と改良Reid法ではバンプの縮小の際のシングルトンの移動回数つまりバンプ縮小の効率に差が出ることがわかる．この差の原因は，前節でも述べたように，改良Reid法ではバンプの最後の列の列シングルトンの移動を優先し，1回のシングルトンの移動で大幅なバンプ縮小を可能としたことにある．

一般に大型疎線形計画問題の係数行列は疎であることと，単体法の初期基底行列は単位行列であることから，バンプ縮小の際に最初と最後の列に列シングルトンが存在する場合が起こる可能性は高いと考えられる．また問題の規模が大きくなれば，そのシングルトンの移動によるバンプの縮小幅も大きくなることが期待される．したがって，大型疎線形計画問題に対して改良Reid法はReid法よりも優れた方法と考えられる．

5．おわりに

本論文では，Reid法のバンプ縮小に関する基本的性質を明らかにし，その性質を十分に生かすためにReid法の列シングルトンの移動順序を変更し，1回の列シングルトンの移動で大幅なバンプ縮小を可能とした改良Reid法を提案した．そして，改良Reid法とReid法のバンプ縮小の際のシングルトン移動回数の関係を明らかにした．

参考文献

1)R.H.Bartels and G.H.Golub: “The Simplex Method of Linear Programming Using LU Decomposition”,Comm. ACM,Vol.12,No.5, pp.266-268(1969).

2)J.J.H.Forrest and J.A.Tomlin: “Updated Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method”,Math.Prog.,Vol.2,pp.263-278(1972).

3)J.K.Reid: “Fortran Subroutines for Handling Sparse Linear Programming Bases”, Report AERE- R.8269,Harwell(1976).

4)J.K.Reid: “A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases”,Math.Prog.,Vol. 24,pp.55-69(1982).

5)V.Chvatal:Linear Programming,W.H.Freeman and Company(1983).

6)M.A.Saunders: “A Fast, Stable Implementation of The Simplex Method Using Bartels-Golub Updating, in Sparse Matrix Computations(J.R.Bunch and D.J.Rose,eds.) Academic Press,pp213-226(1976)