Simplex Method における計算量の減少策について

古瀬大六

序

1947 年に G. B. Dantzig が，LP 問題の最もオーソドックスな解法としての simplex method を発表して以来，この解法についてのすべての問題が解決し尽されてしまったかのように思っている人が少くないようである。然し，事実は全くこれに反する。

この10年間に，gradient method 其他の毛色の変った様々な解法が考案された。然し，それと併行して，simplex method 自身についても，幾多の改善が加えられてきたのであり，その計算量を減らすために，不断の努力が重ねられてきたことを忘れてはならない。

実用的解法，殊に *200×1000* というような大きな問題の解法としては，依然として simplex method が絶対的優位を保持している。それ故，この所要計算量を減らし，同じ型の電子計算機でより大きな問題を扱えるように努めることは，極めて重要なことと言わなければならない。

本論文は，この simplex method における種々の改善策として提案されてきた様々な計算法を，論理的に整理して読者に提示すると同時に，その系統的発展としての，新らしい計算手続きの可能性を探究することを目的とする。

過去10年間における simplex method の改善は，次の方面についてなされてきた。

(1) 各 iteration における計算量減少策

Dantzig が最初に与えた計算法（以下，original method と呼ぶ）は，無駄な計算を多く含んでいた。そこで解答に直接役立たないような数値の計算をで

きるだけ除き，必要な数値を求めるにも最少の計算量ですませるようにと，多くの努力が払われてきた。Dantzig の revised simplex method 或いはその変形としての product form を利用する方法の狙いの一部は，この点にある。

（2） 初期解の求め方

条件式 $Ax=b, (b\geq 0)$ における A が単位行列を含むときは，それを initial basis に選ぶことにより，この問題は簡単に解決される。然らざる場合には，幾つかの人為変数を添加しなければならない。添加された人為変数 (artificial variables) は，最適解の中に含まれてはならないから，計算途中でこれを全部，基底変数の中から追い出しておく必要がある。この追い出し方の巧拙が，全体の iteration の所要回数を大きく支配することになる。

人為変数の追出法として最初に使われたのは，Charnes の M－法，及びそれを電子計算機向きに改めた Orchard-Hays の redundant equation 法（それは，Dantzig によって，revised simplex method の中に受けつがれている），である。この方法を使えば，どのような条件式に対しても，必らず，その人為変数の数と同数の iteration によって，人為変数を含まない basic feasible solution に到達することができる。

但し，この便利な方法にも，一つの避け難い欠点がある。それは，この計算過程 (revised simplex method の Phase I) の狙いが，がむしやらに人為変数を追い出しさえすればよい，という態度であるために，その代りに入ってくる変数が，その目的係数 c の値の大小には全く無関係に選ばれる，という点である。その結果として，追出しの終った時の解の位置が最適解の存在位置から極めて遠く，従って，本来の計算過程 (Phase II) に，必要以上に多くの iterations を要する結果となるのである。

これを防止するには，Phase I と Phase II とを併行して行いさえすればよい。その具体的計算手続きとしては，Wm. Orchard-Hays の Composite Simplex Algorithm, 及び，Dantzig, Ford and Fulkerson の Primal-Dual Algorithm がある。

（3） Degeneracy 対策

Simplex metthod における収束性の保証は，各 iteration における解の basicity（全基底変数が正値をもつ）が保持されることの上に立っている。Basicity が保持されさえすれば，則ち〔$A|b$〕の（$n+1$）個の縦ヴェクトルの中の任意の m 個からなる行列のランクが k であるならば，毎 iteration ごとに目的函数は必らず増加する。従って，有限な解が存在するならば，有限回の繰返しで解に到達する筈である。そのような保証のない場合には，A と b の値をほんの僅かだけ動かすことによって，人為的にそのランクを k に保持する必要がある。Charnes の ω-perturbation method, Dantzig の revised simplex method は，このような対策の代表的なものである。

然し，最近の傾向としては，わざわざ degeneracy 防止機構を計算規則の中に built-in しておかなくても，現実問題で永久循環に陥った例は未だ1例もないのだから，差支えない，という意見が支配的のように見受けられる。Original simplex method 或いは revised simplex method をそのまま使う場合には，上記の2つの方法を採用しても，特にそのために計算量が大量に増加することはない。然し，product form を採用するとなると，rank の維持には特に多くの追加計算を必要とするから，degeneracy 対策を行わないのが普通である。

（4） Dual Method の利用

或る LP 問題が与えられると，必らず，それと等価な dual problem が存在すること，周知の通りである。与えられた問題を primal の形で解くべきか，又は dual の形で解いた方がよいか，は，問題の具体的形態及びその問題に対する吾々の要求の内容如何によって異る。Dual method の方が決定的に有利なのは次2のつの場合である。

（a） Primal を解くには多くの人為変数を必要とするが，dual の場合にはそれを1つも必要としないことがある。則ち，primal の目的函数 $c'x$ の係数 c が全部非負であれば，dual の条件式は $A'u \geqq c$ となり，n 個の slack 変数をそのまま初期解として採用することができる。A が $m \times n$ 行列であるとすれば，一般に $m<n$ であるから，primal で基底に将来組み込まれるべきヴェクトルは，$(m+n)$ から現在の基底ヴェクトル m を除いた残りの n 個であるのに対

し，dual では同じ（$m+n$）個から n 個を除いた僅か m 個が，その選択の対象となるにすぎない。従って，dual の方がより少い iteration で最適解に到達する可能性がある。但し，所要記憶量は逆に増加するから，注意を要する。反対に primal では人為変数を要せず，dual では n 個の人為変数を必要とする場合には，primal 及び dual で選択の対象となるヴェクトルの数は，それぞれ $(n-m)$, $(n+m)$ となって，dual は却って不利である。

(b) 条件式 $Ax=b$ の右辺常数 b が変化した場合に，最適解 x^0 がどう動くか，を精しく知りたいときは，dual で解くべきである。何となれば，LP 問題の諸常数の変化を最も容易に導入できるのは目的函数の変化についてであるから。Primal の条件式の b が，dual では目的係数に変化すること，周知の通りである。

(5) 其他の改善策……新らしいヴェクトルの基底への導入に際して，simplex 法は，各ヴェクトルの単位長に対する目的函数の増減だけを目安にしている。然し，問題は，前の基底から次の基底への移行によってどれだけ目的函数が増すか，ということであるから，通常の simplex criterion δ に，その可能な長さ Δx を乗じた値を以て，導入ヴェクトル決定の基準とすることがよいのではないであろうか。また，初期解を，人為的な，最適解から非常に遠い場所に求めるよりも，直観的判断により，できるだけ最適解に近そうなところから出発する方が能率的であろう。このようにして initial basis を決め，その変数値を求めると，中には負値をとるものもあり得る。これに対しては，revised simplex method の phase I と同じ方法によってこれを追放し，全変数が非負となったところで phase II に切換えればよい。或いはまた，primal-dual method によって更にその繰返し回数を減らすことも可能であろう。

以下，これらの点を，章を分けて詳細に検討してみたい。

第1章　各 iteration の計算量を減らす法

§1・1　Y の代りに B^{-1} を繰越す方法

Dantzig の revised simplex method は，original method に比べて多くの

改善点を含んでいるが，そのうちの重要な改良の1つは，Y という $m\times n$ 行列を毎回修正する代りに，これよりも小さい $m\times m$ 行列 B^{-1} を修正することにより，計算量及び記憶量を減らそうとする試みである。

ここで，original simplex method における Y の 計算手続きを振返ってみよう。*t-th* iteration における Y の値を $Y(t)$ とすれば，次回の $Y(t+1)$ は，次の計算によって求められること，周知の通りである。除去変数 x_{t_r} と導入変数 x_s とは既に決定ずみであるとする。

$$H(t)\cdot Y(t)=Y(t+1) \qquad (1.1.1)$$

この $H(t)$ は，elementary matrix と呼ばれるものであって，次のような簡単な形をもっており，(Yt) の第 S 番目の縦欄 $Y_s(t)\equiv[y_{1s},\cdots\cdots,y_{ms}]'$ から即座に求めることができる。

$$H(t)\equiv\begin{pmatrix} 1 & & -y_{1s}/y_{rs} & & \\ & 0 & & & \\ & & & 0 & \\ & & 1/y_{rs} & & \\ & 0 & & & \\ & & & 0 & 1 \\ & & -y_{ms}/y_{rs} & & \end{pmatrix}$$

則ち，$H(t)$ は，$m\times m$ 単位行列の第 r 番目の縦欄だけを若干修正したものである。

この計算に要する各種演算の量は，次の如くである。

乗　　算	$m\times n$
加 減 算	$(m-1)\times n$
読　　取	$3(m-1)\times n+2m$
書　　込	$m\times n$
記 憶 量	$2(m\times n)$

記憶さるべき数値としては，$Y(t)$ と A とを考えている。A は不必要のようではあるが，途中で誤差の累積を避けるためには，A をも記憶しておく必要があるであろう。

A は通常多くの 0 を含むので，$Y(0)=A$，$Y(1)=H(1)\cdot A$，等々の最初

の Y も多くの 0 を含み，所要計算量もそれに比例して少量ですむ。然し，繰返し回数の増加につれて急速に 0 が減少し，計算量は上記の値に近附く。

200×1,000 というような大型の問題になると，各 iteration の全計算量の大部分がこの Y の計算に費され，その記憶装置の許容限度が，Y の大きさに制約されてしまう結果となる。この計算をいくらかでも縮少することができるならば，より大型の問題を，より速い速度で解くことができるであろう。

そこで，まず，何故に吾々は，毎回毎回このような Y の計算をしなければならないか，を反省してみよう。A を 200×1,000 とすれば，Y もまた 200×1,000 則ち 20萬個の数字からなっているのであるが，実際に必要なのは，そのうちの第 s 番目の縦欄 Y_s だけ，則ち，僅か 200 個の数字だけで，残りの 199,800 個の数字は，全然利用されずに，そのまま次の tableau に引継がれて行くのである。誠にもって，大変な無駄といわなくてはならない。勿論，引継がれた 199,800 個の数字のうちの 200 個は，次の iteration における Y_s の計算に役立つわけであるが，1,000 個のヴェクトルのうち，各 iteration においてどの1個が必要になるかは，その iteration になってみなければ分らないので，必要な1個だけを繰越すというわけにはいかないのである。

然しながら，若し何等かの方法で，Y 自体ではなしに，必要に応じてその任意の縦欄を計算することのできる別のデータを保管しておくことができるならば，Y を使わずにすますことも可能であろう。Y は周知の如く，

$$B(t)\cdot Y(t)=A$$

の解である。従つてそれは，B と A との情報をその中に含んでいる。$B(t)$ は毎回変るけれども，A は始めから終りまで一定である。A は別途に記憶装置の中に記録されているのであるから，それを $Y=B^{-1}A$ の形で一々繰返して書いたり消したりするのは，全くの無駄な操作というべきである。毎回繰越さなければならないのは，Y から A を除いた残りの情報 $B(t)$ 又は $B^{-1}(t)$ だけである。

導入さるべき変数 x_s は，$Y(t)$ が分らなくても，$\delta(t)$ が別途に何等かの方法で計算できさえすれば，

$$\delta_s(t)=\min_{(j,\delta_j<0)}\delta_j(t)$$

として決まる。この s が決まった時に殆めて，記憶装置の中から $B(t)$ と $A_s(t)$ とを取り出し，1つのヴェクトル $Y_s(t)$ を算出すればよい。それは，

$$B(t)\cdot Y_s(t)=A_s$$

という m 元連立一次方程式の解として求められるのであるが，$B(t)$ そのものよりも，その逆行列 $B^{-1}(t)$ を記憶しておき，

$$Y_s(t)=B^{-1}(t)\cdot A_s$$

とした方が，遙かに簡単である（但し A_s は A の第 s 番目の縦欄を表わす）。

この $B^{-1}(t)$ から，次の iteration における $B^{-1}(t+1)$ を求める計算は，Y の場合と全く同じである，則ち：

$$H(t)\cdot B^{-1}(t)=B^{-1}(t+1)$$

により，次の iteration への変換が行われる。このように，Y の代りに B^{-1} を記録修正する方法の利点は次の通りである。

（1） $Y(m\times n)$ の代りに，遙かに小さい $B^{-1}(m\times m)$ の行列を計算，記録するだけですむ。

（2） Original methodで Y だけを記憶し，A は記憶しない場合でも，Y はその大部分が非零であるのに，A の方は極めて多くの 0 を含むので，その全記憶量は，original method（Y を記憶）よりも revised method（B^{-1}, A を記憶）の方が多くの場合少量の記憶量ですむ。

此所に，$B^{-1}(t)$ から $B^{-1}(t+1)$, $Y_s(t+1)$ を計算するに要する計算量を記しておく。

乗　算	$2m^2$
加減算	$(m-1)(2m-1)$
読　取	$m(5m+1)$
書　込	$m(m+1)$
記　憶	$m(m+n)$

これを originad metthod の場合と，m=200, $n=1000$ の問題に就いて比較してみよう。

	Original	Revised
乗　算	$200K$	$80K$

加減算	199K	79K
読　取	597K	200K
書　込	200K	40K
記　憶	400K	240K

全体を通観して，約半分の計算量及び記憶量で済むものと判断される。この revised method の相対的有利性は，行列 A が正方形に近いほど小さく，横に長いほど大となる。但し，上記の比較は，Y と，B^{-1} 及び Y_s との比較であり，simplex criterion δ の計算量の比較を含まない。original method において，Y から δ を求める計算は，d を基底変数に対する c の 部分ヴェクトルとすれば，

$$\delta'=d'Y-c'$$

に従って計算される。他方，$Y=B^{-1}A$ であることを想い出せば，

$$\delta'=d'B^{-1}A-c'$$

となるから，$B^{-1}(t)$ が分っていれば，これから δ を計算できる筈である。然し，δ の値を，上の式からまともに計算したのでは，大変な手間がかかる。

$d'B^{-1}$ は m 個の要素からなるヴェクトルであり，従って $B'u=d$ の解 u に外ならない。それはまた，B を最適基底とする dual の解，則ち shadow prices と解釈することができる。この $u(t+1)$ と $u(t)$ との間には，次のような関係式が成り立つ。

$$\begin{aligned}
u'(t+1)&=d'(t+1)B^{-1}(t+1)\\
&=d'(t+1)H(t)B^{-1}(t)\\
&=\left[d_{t_1},\cdots,d_{t_{r-1}},-d'(t)\frac{Y_s(t)}{y_{rs}(t)}+c_s+\frac{c_s}{y_{rs}(t)},d_{t_{r+1}},\cdots,d_{t_m}\right]B^{-1}(t)\\
&=\left[d_{t_1},\cdots,d_{t_{r-1}},-\delta_s(t)/y_{rs}(t)+c_s,d_{t_{r+1}},\cdots,d_{t_m}\right]B^{-1}(t)\\
&=d'(t)B^{-1}(t)+\left[0,\cdots,-\delta_s(t)/y_{rs}(t),\cdots,0\right]B^{-1}(t)\\
&=u'(t)+\left[0,\cdots,0,-\delta_s(t)/y_{rs}(t),0,\cdots,0\right]B^{-1}(t)
\end{aligned}$$

従って，この u の変換計算を B^{-1} のそれと一括して記せば，

$$\left(\begin{array}{c|c} 1 & 0\cdots-\delta_s/y_{rs}\cdots 0 \\ \hline 0 & H(t) \end{array}\right)\cdot\left(\begin{array}{c|c} 1 & u'(t) \\ \hline 0 & B^{-1}(t) \end{array}\right)=\left(\begin{array}{c|c} 1 & u'(t+1) \\ \hline 0 & B^{-1}(t+1) \end{array}\right)$$

となる。この左辺第1項の行列を $\overline{H}(t)$，第2項の行列を $\overline{B}^{-1}(t)$ と書けば，更に簡単化されて，

$$\overline{H}(t)\cdot\overline{B}^{-1}(t)=\overline{B}^{-1}(t+1)$$

となり，B^{-1} と u とが変換行列 $\overline{H}$ によって一挙に変換されることになる。斯くして求められた $u(t+1)$ に，

$$\delta'(t+1)=u'(t+1)A-c'$$

なる演算を施せば，目的とする $\delta(t+1)$ が計算される。

この δ の計算に要する演算量を，original method と revised method とについて比較すれば，m 個の基底ヴェクトルの δ は必らず 0 であるから改めて計算しないこととして，Phase II では，

	original	revised
乗　算	$m(n-m)$	$m(n-m)+m$
加減算	$m(n-m)$	$m(n-m)+(m-1)$
読　取	$3m(n-m)$	$3m(n-m)+3m$
書　込	$(n-m)$	$(n-m)+m$
記　憶	0	m

となり，revised method の方がいくらか余計な計算を必要とするが，全体から見て殆んど無視できる程度の違いにすぎない。

最後に，revised simplex method における δ，B^{-1}，Y_s，$v(=B^{-1}b)$ 及び $\pi(=d'v)$ の変換計算過程を一括して示しておく。

$$H(t)\cdot\left[\begin{array}{c|c}\pi(t)\\ v(t)\end{array}\middle|\ \overline{B}^{-1}(t)\right]=\left[\begin{array}{c|c}\pi(t+1)\\ v(t+1)\end{array}\middle|\ \overline{B}^{-1}(t+1)\right]$$

$$\overline{B}^{-1}(t+1)\left[\begin{array}{c}-c'\\ A\end{array}\right]\rightarrow\left[\begin{array}{c|c|c}\multicolumn{3}{c}{\delta'(t+1)}\\ \hline 0 & Y_s(t+1) & 0\end{array}\right]$$

第2式は，δ の計算には A の全体を使い，Y_s の計算には A_s だけを使うことを意味する。

以上の比較検討により，revised simplex method は，全計算過程について考えても，original simplex method に比べて，極めて少量の演算及び記憶量ですむことが明らかとなった。その違いは，A が横長であればあるほど，また，

A の中に含まれる 0 の割合が大であるほど，大きい。

§ 1・2 Product Form of the Inverse B^{-1}

計算時間を減らす試みの基本的な着眼点は，直接利用されないデータの計算を極力避け，その必要性がはっきりしたときにのみ，それを計算する，という態度をとることである。

吾々は既に，B^{-1} を計算することによって Y を計算する手数を省いた。更に B^{-1} についてもまた，同様の省略ができないだろうか。$B^{-1}(t)$ は，

$$H(t-1)\cdot H(t-2)\cdot\cdots\cdots\cdot H(1)=B^{-1}(t)$$

の手続きによって求められたものである。従って，$B^{-1}(t)$ を記憶する代りに，$H(1)$, $H(2)$,……を記憶しておき，必要に応じてそれらを掛け合せることにしても差支ない筈である。問題は，その何れが，より有利であるか，という点である。行列を幾つも掛け合わせる計算は尨大な時間と手数を要するように想像されるかもしれない。然し，実際の計算手続きは，極めて僅かの計算量ですむ。

まず，$H(0)$, $H(1)$,…,$H(t-1)$ から $u'(t)$ を計算する場合を考えてみよう。

$$u'(t)=u'(0)B^{-1}(t-1)$$
$$=u'(0)\,E(t-1)E(t-2)\cdots\cdots E(0)$$

この計算は，E の累乗を先に計算するよりも，左から順に，$u'(0)E(t-1)$, $\{u'(0)E(t-1)\}E(t-2)$ と進めて行く方が簡単である。但し，$u'(0)$ は，その第1項だけが1，其他は m 個の 0 要素からなるヴェクトルである。E は何れも単純な elementary matrix であるから，このヴェクトルと E との乗算1回につき，乗算 m 回，加算 $(m-1)$ 回，読取 $(m+1)$ 回ですむ。E は，その非零縦欄の番号と，その m 個の要素の値とを記録しておけば充分である。E に掛けられるヴェクトルの値は，計算用高速ドラムの中にそのまま残しておけばよく，一々記録，読取の必要はない。

また，$Y_s(t)$ の計算は，

$$Y_s(t)=B^{-1}(t-1)\,A_s$$
$$=E(t-1)\cdot E(t-2)\cdots\cdots E(0)\cdot A_s$$

であり，これもまた，$E(0)A_s$，$E(1)\{E(0)\cdot A_s\}$，$E(2)\,[E(1)\{E(0)\cdot A_s\}]$ の順に，行列×ヴェクトルの計算を n 回くりかえせばよい。その1回当りの所要計算量は，$u(t)$ 計算の場合と全く同じである。

従って，B^{-1} を計算する場合と，その product form を利用する場合とにつき，t-th iteration における Y_s と u との所要計算量を比較すれば，$t<m$，且つ，一旦導入されたヴェクトルが再び除れることはないと仮定して，

	B^{-1}法	H法
乗　算	$3tm$	$2mt$
加減算	$3t(m-1)$	$2(m-1)t$
読　取	$7tm-3t$	$2(m+1)t$
書　込	$m(t+2)$	$3m+1$
記　憶	$(m+1)t$	$(m+1)t$

この表から明かなように，$t<m$ なる範囲内では，H 法の方が絶対的に優れている。然し $t=m$ となり，更に t が m を超えると，B^{-1} 法の所要計算量は，上の表で $t=m$ とおいた値に固定されるのに反し，H 法では，t に比例してどんどん増し続ける。その限界点は，乗算と加減算とについては，

$$\begin{cases} 3m^2 = 2mt \\ 3m(m-1)=2t(m-1) \end{cases}$$
$$\therefore\quad t=\frac{3}{2}m$$

則ち，$1.5m$ 回を超えるならば，反対に B^{-1} 法の方が有利となる。従つて $1.5m$ 回に近附いたならば，B^{-1} 法に切換えた方が有利であろう。

この product form の利点として，記憶量の減少を挙げる人もあるが，この点では，$t<m$ なる限り，B^{-1} 法と H 法との間の優劣の差はない。t が m を超えれば，H 法の方が却つて不利である。

読取回数でも，t が大体 $3m$ 以下であれば H 法の方がよく，書込回数では，t の如何に拘わらず H 法の方が遙かに優れている。一方は，毎回毎回1行づつ数字のつまって行く $m\times m$ 行列を，消しては書き，消しては書きしなけ

ればならぬのに反し，product form では，僅か $m+1$ 個の数字（則ち，1行分の数字）を記録するだけで足り，これに u と Y_s との値を入れたとしても，$(3m+1)$ 個で足りるわけである。

§1・3 Original Method の Product Form による改善案

Revised method の提案以来，original method は非能率的な計算法として忘れ去れようとしている。然し，original method も，そう棄てたものではない。なるほど，revised method は，Y の代りに B^{-1} を繰返し修正することによって，計算量と記憶量とを大幅に節約することができた。然し，その反面，δ の計算量を節約する可能性を放棄する結果となったのである。

Y を毎回計算することは，決して，全然無駄なことではない。δ の計算は，original method では，

$$\delta' = d'Y - c'$$

であり，revised method では

$$\delta = u'B^{-1} - c'$$

である。後者の方が，いくらか多くの計算（u を計算するための計算量）を必要とすることは，§1・1の終りで説明しておいた通りであるが，他の面における大幅の節約効果をふいにしてしまうほどのものではない。

それよりも遙かに重大なのは，Y を使えば，$\delta'(t+1)=\delta'(t)-[0,\cdots,0,-\delta_s(t)/y_{rs}(t),0,\cdots,0]\,Y(t)$ によって，僅かに $(n-m)$ 回の乗算と，同じ回数の加減算とで δ を計算できる便利な方法が使えるのに，revised method ではこのような巧妙な方法を使えない，という点である。

Original method における δ の計算法は，$\delta(t+1)$ の計算に際して，前回計算された $\delta(t)$ の値についての情報を全然利用していない。$\delta(t)$ の中には，$c,d(t)$，$Y(t)$ についての情報が含まれている筈であり，且つ，次期の $c,d(t+1)$，$Y(t+1)$ は，前記のそれを部分的に修正したものにすぎないのであるから，$\delta(t+1)$ を基礎データから計算し直すよりも，δ の変化分だけを計算して，これを $\delta(t)$ に加えることによって $\delta(t+1)$ を求めた方が，遙かに少量の計算ですむに違いない。

そこで，上記の original simplex method における δ_j の計算手続きを振返つてみると，次の如く表わすことができる。則ち，$Y(t)$ の j 番目の縦欄を $Y_j(t)$，新たに基底ヴェクトルに追加さるべきヴェクトル P_s に対する $Y(t)$の縦欄を $Y_s(t)$:

$$Y_j(t)\equiv\begin{pmatrix} y_{1j} \\ \vdots \\ y_{rj} \\ \vdots \\ y_{mj} \end{pmatrix}, \qquad Y_s(t)\equiv\begin{pmatrix} y_{1s} \\ \vdots \\ y_{rs} \\ \vdots \\ y_{ms} \end{pmatrix}$$

とし，除去さるべき基底変数 x_{t_r} の位置が，$Y(t)$ の上から r 番目の横欄に相当するものとすれば：

$$Y_j(t+1)=Y_j(t)-Y_s(t)\frac{y_{rj}}{y_{rs}}+I_r\left(\frac{y_{rj}}{y_{rs}}\right)$$

但し，$I_r(x)$ は r 番目が x，其他は全部 0 要素からなる縦ヴェクトルを表わす。次に，この $Y_j(t+1)$ に $d(t+1)$ を乗じて $z_j(t+1)$ を求めるのであるが，上式の右辺にも同じ $d(t+1)$ を乗ずれば：

$$z_j(t+1)=d'(t+1)Y_j(t+1)=d'(t+1)\left\{Y_j(t)-Y_s(t)\frac{y_{rj}}{y_{rs}}\right\}+d'(t+1)I_r\left(\frac{y_{rj}}{y_{rs}}\right)$$

上の式において，右辺の $\{Y_j(t)-Y_s(t)y_{rj}/y_{rs}\}$ は，その第 r 番目の要素は 0 であるから，それに $d'(t+1)$ を乗じても，また $d'(t)$ を乗じても，その積の値は同じであり，更に，$I_r(y_{rj}/y_{rs})$ に $d'(t+1)$ を乗ずることは c_s に y_{rj}/y_{rs} を乗ずることに外ならないから：

$$z_j(t+1)=d'(t)\left\{Y_j(t)-Y_s(t)\frac{y_{rj}}{y_{rs}}\right\}+c_s\frac{y_{rj}}{y_{rs}}$$

$$=z_j(t)-z_s(t)\frac{y_{rj}}{y_{rs}}+c_s\frac{y_{rj}}{y_{rs}}$$

$$=z_j(t)-\frac{y_{jr}}{y_{rs}}\left\{z_s(t)-c_s\right\}$$

$$=z_j(t)-\frac{y_{rj}}{y_{rs}}\delta_s(t)$$

となる。つまり，$z_j(t+1)$ を求めるのに，$Y_j(t+1)$ と $d(t+1)$ とを掛け合

わせる(m 回の乗算と $m-1$ 回の加算とを要する) 代りに，既に計算済みの $\delta_s(t)$ に y_{rj}/y_{rs} を乗じたものを，これも既に計算済みの $z_j(t)$ から，差引くだけで，(乗算1回，除算1回，加減算1回)，同じ結果に到達できることが分つた。従つて $\delta(t+1)$ は：

$$z_j(t+1)-c_j = z_j(t)-c_j - \frac{y_{rj}}{y_{rs}}\delta_s(t)$$

$$\therefore\quad \delta_j(t+1)=\delta_j(t)-\frac{y_{rj}}{y_{rs}}\delta_s(t)$$

この y_{rj}/y_{rs} は，$Y(t)$ の r 番目の横欄の要素であるから，$\delta(t+1)$ を計算する際には，既に計算済みの与えられた値であり，従つて上記の計算は，僅かに1回の乗算と1回の減算とですむことになり，大幅な計算量が節約される。

Original method の利点は，この δ 計算の便利さだけでなく，次のような極めて単純な行列演算の形に定式化できる，という点にある。

$$\left(\begin{array}{c|c} 1 & 0\cdots-\delta_s(t)/y_{rs}(t)\cdots 0 \\ \hline 0 & H(t) \end{array}\right)\cdot\left(\begin{array}{c|c} \pi(t) & \delta'(t) \\ \hline v(t) & Y(t) \end{array}\right)$$

$$=\left(\begin{array}{c|c} \pi(t+1) & \delta'(t+1) \\ \hline v(t+1) & Y(t+1) \end{array}\right)$$

この一回の演算で，rs の決定を除いたすべての計算が一挙に遂行されるのであるから，計算機のプログラミングも極めて簡単なものになるに違いない。その各 iteration 当りの所要計算量は，

乗　算	$(n+1)(m+1)$
加減算	$(n+1)\times m$
読　取	$3(n+1)(m+1)-(n+1)$
書　込	$(n+1)\times(m+1)$
記　憶	$(n+1)\times(m+1)$

それにも拘らず，revised method によってその地位を奪われるに到つた理由の一つは，前にも述べたように，$Y(t)$ の計算量及び記憶量があまりにも多す

ぎる，という欠点によるものであった。

$Y(t)$ の中で，直接必要なデータは，その第 s 番目の縦欄 $Y_s(t)$ と，r 番目の横欄 $Y_r(t)$ とだけであり，其他の数値は，後の iteration における利用可能性を予想して，何時かは役立つかもしれないという見込みで計算されているにすぎない。そこで，revised method の場合と同様に，product form を使い，必要な数値以外は計算しない，という方針を採ることにより，その計算量・記憶量を減らすことができるであろう。 則ち，

$$\pi(0)=d'(0)b$$

$$v(0)=b$$

$$\delta'(0)=-c'$$

$$Y(0)=A$$

であるから，

$$\overline{H}(t)\cdots\cdots\overline{H}(0)\left(\begin{array}{c|c}\pi(0) & -c' \\ \hline b & A\end{array}\right)=\left(\begin{array}{c|c}\pi(t+1) & \delta'(t+1) \\ \hline v(t+1) & Y(t+1)\end{array}\right)$$

という product form に改められる。このうち，$\pi(t+1)$ と $v(t+1)$ とは，$\overline{H}(t)$ と $\pi(t)$, $v(t)$ とから簡単に計算できるので，product form に改めることは却つて不利益である。

$\delta'(t+1)$ の計算は，

$$\overline{H}(t)\cdots\cdots\overline{H}(0)\left(\begin{array}{c}-c' \\ \hline A\end{array}\right)\rightarrow\left(\begin{array}{c}\delta'(t+1) \\ \hline \end{array}\right)$$

に従つて計算されるわけであるが，この $\overline{H}$ の累乗のうち，上記の計算に必要なのは，その最上横欄だけである。それは，

$$B^{-1}(t)=\overline{H}(t)\cdots\cdots\overline{H}(0)$$

の最上横欄であるから，結局，revised method で計算された $u(t)$ に外ならないことは，容易に理解されるであろう。それ故，δ の計算に関する限り，original method に product form を加味することは，その計算量を減らすことには役立たない。

$\delta(t+1)$ を有効に計算するには，それ故，u という価格ヴェクトルを使わずに，$\delta(t)$ と $Y_r(t)$ とからこれを求めるように努めなければならない。$\delta(t)$ は既に与えられているから，残りの $Y_r(t)$ を product form で計算する方法を考えてみよう。

$$\overline{H}(t)\cdot\cdots\cdots\overline{H}(0)\begin{bmatrix} A \end{bmatrix}\rightarrow\begin{bmatrix} 0 \\ \hline Y_r(t+1) \\ \hline 0 \end{bmatrix}$$

であり，$\overline{B}^{-1}(t)=\overline{H}(t)\cdot\cdots\cdots\overline{H}(0)$ の上から $(r+1)$ 番目の横欄のヴェクトルと行列 A との乗算によって $Y_r(t+1)$ が求められる。この，$\overline{B^{-1}}(t)$ の第 $(r+1)$ 横欄を求めるための計算量は $u(t)$ の計算量と全く同じであり，それに Aを乗じて $Y_r(t+1)$ を算出するための計算量は，u と A とから Zを求める計算量と同じであり，最後に $Y_r(t+1)$ と $\delta(t)$ とから $\delta(t+1)$ を求める計算量は，Z と c とから δ を求める計算量よりも $(n-m)$回多くの乗算を必要とする。従つて，$\delta(t+1)$ の計算量を減らす目的で $Y_r(t+1)$ を求めることは，却つてその計算量を僅かながらふやす結果となる。それ故，$\delta(t+1)$ の計算は，$\overline{B^{-1}}(t)$ の第1行 u と A との積からこれを求めることにした方がよい。

最後に，$Y_s(t+1)$ は，

$$\overline{H}(t)\cdot\cdots\cdots\overline{H}(0)\cdot A_s = Y_s(t+1)$$

として求められるが，これは revised method に product form を利用した場合と，全く同じになる。

以上，吾々は，original method を product porm によって改善することを試みたのであるが，結果においては，revised method に product form にを採用した場合と，全く同一の計算手続きに到達したわけである。

第2章　初期解の求め方

§2・1 Original and Revised Methods はおける初期解の求め方

過去の経験或いは直観的判断に基いて初期解を求めることができれば，それは通常，最適解からあまり遠くない位置にある場合が多いので，iteration の所

要回数を大幅に減らすことができる。

多くの場合には，然し乍ら，全然見当が附かないか，又は，或る程度の見当をつけることができても，それを苦労して解いてみたら，幾つかの変数が負値をとってしまったという結果になる。

そこで，本格的な LP 解法においては，全く形式的な手続きだけで初期解を求めようと試みるのが常である。そのためには，スラック変数と人為変数（artificial variables）の両者が使われる。

（1） 条件式の右辺常数 b の符号を全部非負に改める。その結果，不等号の向きは不揃いになる可能性がある。

（2） 不等号が右向き（$\leq b$）の式の左辺には係数1のスラック変数を加え，左向き（$\geq b$）の式の左辺には係数（-1）のスラック変数を追加することによって，全部を条件等式に改める。これらスラック変数の値は，目的函数とは無関係であるから，それに対する目的係数 c は，何れも 0 とおく。

（3） このようにして得られた条件式 $Ax=b\,(b\geqq 0)$ の係数行列 $A(m\times n)$ の中から，m個のヴェクトルを選ぶことによって，単位行列を取り出すことができるならば，それに対応する x の値を b，其他の x を 0 とおくことによって，初期基底許容解を簡単に得ることができる。

（4） それが不可能であれば，則ち，A の中の互に独立な単位ヴェクトルの数が m に足りないときは，その不足分を更に追加することによって，無理にでも単位行列をこしらえあげなければならない。これらの追加変数（人為変数）は，等式の左辺に無理に割り込んだものであるから，最終解のなかに残されては困る。これらの人為変数を，最適解への途中で全部追い出すためには，問題が最大問題であれば，その目的係数を $-M$（Mは他の如何なる有限数値よりも大な常数）とし，最小問題であれば M とすることによって，自動的に遂行される。

（4′） 電子計算機を使用するときは，上記の M 法と原理的には全く同じであるが，Beale の redundant equation を利用するのが便利である。それは，追放さるべき l 個の人為変数の総和にマイナス符号を附けた値を表わす今一つの人為変数 x_{n+1} を導入して，この x_{n+1} の値を最大にすることを目的として繰返

し計算を遂行し，この x_{n+1} が負から 0 に転ずることによって人為変数全部の追放を完了する方法である。則ち，

$$x_{n+1}+x_{n+2}+\cdots\cdots+x_{n+1+l}=0$$

を，$(m+1)$ 番目の，或いは 1 番目の条件式として追加する。但し，このままでは人為変数の単位ヴェクトル性を失わせることになるので，この式と，今迄の m 個の条件式の両辺にマイナス符号を附けた式とを辺々相加えて，

$$a_{01}\ x_1+\cdots\cdots+a_{0n}\ x_n+x_{n+1}\qquad\qquad=b_0$$

但し，

$$a_{0j}=-\sum_{i=1}^{m}a_{ij}$$

$$b_0=-\sum_{i=1}^{m}b_i$$

と変形するならば，x_{n+1}，$x_{n+2},\cdots,x_{n+1+l}$ の $(l+1)$ 個の人為変数と，A の中に含まれている $(m-l)$ 個の変数，合計 $(m+1)$ 変数を基底変数として選ぶことができる。但し x_{n+1} だけは負値をとる。revised simplex method の Phase I の計算がこれに外ならない。

この (4)，(4′) の計算は，単に初期解を探すためのもので，本来の目的函数の最大値を探すという狙いには全く無縁なものであり，出来ればこれを行わずにすましたい計算である。その 1 つの対策として，人為変数を，如何なる場合にも，僅か 1 個に減らしてしまう方法が考えられている (Gass による)。それには，スラック変数を加えて条件等式　$Ax=b$ $(b\geqq 0)$ を作り上げた上で，その中からマイナス符号のついた単位行列をもつ式（則ち，人為変数の追加を必要とする式）だけを抜き出す。この選ばれた式のうちで，右辺常数 b が最大のものを除いて，残りの両辺にマイナス 1 を乗ずることによって，マイナス単位行列をプラス単位行列に転化させる。但しこれでは右辺がマイナスになるから，これをプラスにするために，先程残しておいた $max\ b_i$ をもつ式を，これらに辺々相加える。加えられる方の式の右辺 $-b$ の絶対値は，これに加える $max\ b_i$ よりも小であるから，この結果，右辺は必らず非負となり，従つてこれらの式の中の単位ヴェクトルを初期解として採用することができる。従つて，$max\ b_i$ を右辺に持つ唯一つの式に対して，唯一つの人為変数を導入するだけ

で，初期解を求めることができる。

この方法は，一見，極めて有利な初期値発見法であるかのようであるが，重大な一つの欠点をも伴せ持つていることを忘れてはならない。それは，係数行列 A が多数の 0 を含む場合，上記の手続きのためにその特性を失い，Aの非零要素が大幅に増大する危険を伴うことである。従つて，A の大部分が非零要素である場合を除き，その採用に当つては慎重でなければならない。

§ 2・2 変数に假想的上限を附加する方法

次に説明する初期解の求め方は，Wagner の考案になるものである。但し，彼の提案は，dual method に関するものであるので，ここではそれを primal のままで扱う方法を考えてみよう。

(1) 条件式 $Ax \leqq b$ のうち，等式についてはその右辺を非負に改めた上で人為変数を加え，その目的係数を $-M$ とする。残りの不等式にはスラック変数を加えて等式化し，その目的係数を 0 とおく。これにより，新らしい条件式 $Ax=b$, $x \geqq 0$ が得られる。

(2) その結果，$Ax=b$ の右辺定数 b が全部非負 ($b \geqq 0$) であれば，上の m 個の追加変数を b，其の他の変数を 0 とおくことにより，初期許容解が得られる。

(3) b が若干の負値を含むときは，それに対応する追加変数 $\overline{x}$に，或る假想的な極めて高い上限 を 与え，$\overline{x}$ と d との差額を $\overline{x'}$ とする。則ち，

$$\overline{x} + \overline{x'} = d$$

$$\therefore \quad \overline{x} = d - \overline{x'}$$

(4) 目的函数及び条件式中の $\overline{x}$ に，この $d-\overline{x'}$ を代入し Aa を右辺に移項すれば，$\overline{x'}$ に対する係数は負の単位ヴェクトル，その右辺常数は負となり，従つて，$-\overline{x} = b - d$, 残りの追加変数は $x = b$, とおくことによって，初期非負許容解が求められる。

簡単な計算例を示しておこう。Primal における条件式が，次のような値を示すものとする。

$$max \quad -x_1 + 2x_2 - x_3 + 3x_1$$

$$\begin{cases} x_1 + 3x_3 + 4x_4 \leq 30 \\ -2x_1 + x_2 + x_4 \leq -50 \\ -2x_3 + 2x_4 \leq -20 \\ -2x_2 - x_4 = -5 \end{cases}$$

これを(1)の手続きによって等式化すれば，

$$max \quad -x_1 + 2x_2 - x_3 + 3x_4 - Mx_8$$

$$\begin{cases} x_1 + 3x_3 + 4x_4 + x_5 = 30 \\ -2x_1 + x_2 + x_4 + x_6 = -50 \\ -2x_3 + 2x_4 + x_7 = -20 \\ 2x_2 + x_4 + x_8 = 5 \end{cases}$$

このうち，x_6 と x_6 とに，

$$x_6 = -x'_6 + d_6$$

$$x_7 = -x'_7 + d_7$$

を代入すれば，

$$max \quad -x_1 + 2x_2 - x_3 + 3x_4 - Mx_8$$

$$\begin{cases} x_1 + 3x_3 + 4x_4 + x_5 = 30 \\ -2x_1 + x_2 + x_4 - x'_6 = -50 - d_6 \\ -2x_3 + 2x_4 - x'_7 = -20 - d_7 \\ 2x_2 + x_4 + x_8 = 5 \end{cases}$$

$$x_1, \cdots, x_8 \geqq 0$$

従つて，その初期解は，

$$\begin{cases} x_5 = 30 \\ x_6 = 50 + d_6 \\ x_7 = 20 + d_7 \\ x_8 = 5 \end{cases}$$

で与えられる。

この方法の利点は，Gass の場合と異り，A の 0 要素をそのまま保持できることである。d の値さえ上手に決めることができるならば，極めて優れた方法といつてよいであろう。

§ 2・3 Dual の初期解を利用する方法

Primal の初期解に多くの人為変数を必要とするときは，その dual を一応検討してみるのが常道である。則ち，primal

$$max\ c'x$$
$$Ax = b \geqq 0$$
$$x \geqq 0$$

において，$c \leqq 0$ であるならば，その dual,

$$min\ b'u$$
$$A'u \geqq c$$

の初期解は，条件式に n 個のスラック変数を追加するだけで，それを $-c$ とおくことによって求められる。

$c \leqq 0$ でない場合でも，Wagner の方法でこれを $c \leqq 0$ に改めることが可能である。則ち，$c > 0$ なる目的係数をもつ変数 $\overline{x}$ だけについて假想的上限 d を設定し，

$$\overline{x} = -\overline{x'} + d$$

を条件式と目的函数に代入すれば，$\overline{x}$ の目的係数は，

$$c'\overline{x} = -c'\overline{x'}$$

となり，その目的を達することができる。

また，Dantzig, Ford and Fulkerson の primal-dual algorithm では，やはり一種の假想的上界 x_1 を全変数の合計に対して加え，

$$x_1 + \cdots\cdots + x_n \leq b_0$$
$$\therefore\ x_0 + x_1 + \cdots\cdots + x_n = b_0$$

これと，条件式 $Ax=b$ とを連立させた primal を考える。

この修正された primal に対する dual は，

$$mim\ b_0 u_0 + b_1 u_1 + \cdots\cdots + b_m u_m$$
$$\begin{cases} u_0 + a_{11} u_1 + \cdots\cdots + a_{m1} u_m \leq c_1 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ u_0 + a_{1n} u_1 + \cdots\cdots + a_{mn} u_m \leq c_n \end{cases}$$

（但し u_0 , u_1 , …, u_m の符号は任意）

となる。ここで，$u_0 = min\, c_t$, $u_1 = \cdots\cdots = u_n = 0$ を dual の初期解に選ぶ。これは feasible solution ($A'u \leq c$) ではあるが，basic ではない。そこで，これと complementary な primal を考え，その infeasible ($Ax \neq b$, $x \geq 0$) basic solution における infeasibility を除くような iterative algorithm を行えば，infeasibility の除去と同時に，optimal solution が得られる。

（以下次号）