Sec3.3:Implementation of the simolex method

この章ではシンプレクス法 (単体法) を実行するいくつかの方法 (2 つ) について説明する。アルゴリズムの概要から、ベクトル $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ が重要な役割を果たすことは明らか。このベクトルが利用できる場合、被約費用、移動の方向、ステップサイズ θ^* は簡単に計算できる。したがって、2 つの実装法の主な違いは、ベクトル $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ の計算方法と、ある反復から次の反復に運ばれる関連情報の量にある。
異なる実装法を比較するときは、$(m \times m)$ 行列 $\boldsymbol{B}$、ベクトル $\boldsymbol{b} \in \mathbb{R}^m$ が与えられた時、$\boldsymbol{B}$ の逆行列の計算、$\boldsymbol{Bx} = \boldsymbol{b}$ 形式の線形システムを解くには $\mathcal{O}(m^3)$ の計算量、行列-ベクトル積 $\boldsymbol{Bb}$ の計算には、$\mathcal{O}(m^2)$ の計算量、2 つの m 次元ベクトルの内積 $\boldsymbol{p}'\boldsymbol{b}$ を計算するには、$\mathcal{O}(m^2)$ の計算量が必要である、ということに注意することが大切。(Sec1.6 参照)

Naive implementation

最初に、ある反復から次の反復に補助情報が伝達されない最も単純な実装法について説明する。反復の開始時に、現在の基底変数の添字 $B(1), ..., B(m)$ がある。基底行列 $\boldsymbol{B}$ を作成し、未知のベクトル $\boldsymbol{p}$ について線形システム $\boldsymbol{p}'\boldsymbol{B} = \boldsymbol{c}'_B$ を解くことで、$\boldsymbol{p}' = \boldsymbol{c}'_B\boldsymbol{B}^{-1}$ が得られ、ある変数 x_j の被約費用 $\bar{c_j} = c_j - \boldsymbol{c}'_B\boldsymbol{B}^{-1}\boldsymbol{A}_j$ は

$$\bar{c_j} = c_j - \boldsymbol{p}'\boldsymbol{A}_j \tag{1}$$

で表せる。採用するピボットルールに応じて、被約費用をすべて計算してもよいし、負の被約費用を持つ変数に出会うまで 1 つずつ計算してもよい。ベクトル $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$ を決定するため、基底となる列 $\boldsymbol{A}_j$ を選択し、線形連立方程式 $\boldsymbol{Bu} = \boldsymbol{A}$ を解く。この時点で、現在の基本実行可能解からの移動方向を決めることができる。最終的に θ^* と、基底から外す変数を決定し、新しい基本実行可能解を構築する。

システム $\boldsymbol{p}'\boldsymbol{B} = \boldsymbol{c}'_B$ や $\boldsymbol{Bu} = \boldsymbol{A}$ を解くには $\mathcal{O}(m^3)$ の計算量が必要であり、さらに、すべての変数の被約費用を計算するには、非基底列 $\boldsymbol{A}$ のそれぞれとベクトル $\boldsymbol{p}$ の内積を形成する必要があるため、$\mathcal{O}(mn)$ の計算量が必要です。反復ごとの合計計算量は $\mathcal{O}(m^3 + mn)$ である。(この後紹介する実装法では、$\mathcal{O}(m^2 + mn)$ の計算量で実装可能であることがわかる。) したがって、ここで説明した実装法は、一般的にかなり非効率的である。一方、特殊な構造に関する特定の問題の場合、線形システム $\boldsymbol{p}'\boldsymbol{B} = \boldsymbol{c}'_B$ や $\boldsymbol{Bu} = \boldsymbol{A}$ の計算は非常に高速である。このような場合は、この実装法は実用上有効。シンプレックス法をネットワークフローの問題に適用するときは、第 7 章でこの点を再検討する。

Revised dimplex method

単純な実装法での計算負荷のほとんどは 2 つの線形連立方程式を解く必要からきている。次の実装法では、行列 $\boldsymbol{B}^{-1}$ は各反復の開始時にすでに求められており、ベクトル $\boldsymbol{c}'_B\boldsymbol{B}^{-1}$ や $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ は、行列-ベクトル積によって計算できる。このアプローチを実用的にするには、基底変換をするたびに行列 $\boldsymbol{B}^{-1}$ を更新する効率的な方法が必要である。この方法を説明する。まず、

$$\boldsymbol{B} = [\boldsymbol{A}_{B(1)} \cdots \boldsymbol{A}_{B(m)}] \tag{2}$$

を反復開始時の基底行列とし、

$$\bar{\boldsymbol{B}} = [\boldsymbol{A}_{B(1)} \cdots \boldsymbol{A}_{B(l-1)} \quad \boldsymbol{A}_j \quad \boldsymbol{A}_{B(l+1)} \cdots \boldsymbol{A}_{B(m)}] \tag{3}$$

を次の反復の開始時の基底行列とする。これらの 2 つの基底行列は、l 番目の列 $\boldsymbol{A}_{B(l)}$ が A_j に置き換えられている以外は、同じ列を持っている。その場合、$\boldsymbol{B}^{-1}$ には、$\bar{\boldsymbol{B}}^{-1}$ の計算で利用できる情報が含まれていると予想できる。これは実際に正しいことが証明できる。(演習 3.13 参照)

定義 3.4

行列が与えられた場合、必ずしも正方形である必要はなく、ある行に同じ行もしくは別の行の定数倍を追加する操作は、elementary row operation(行基本変形) と呼ばれる。

次の例は、$\boldsymbol{Q}$ を適切に構築された正方行列とした時、行列 $\boldsymbol{C}$ で行基本変形を実行することは、行列 $\boldsymbol{QC}$ を形成することと同等であることを示している。

Ex3.3

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad \boldsymbol{C} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad , \quad \boldsymbol{QC} = \begin{bmatrix} 11 & 14 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \tag{4}$$

行列 $\boldsymbol{C}$ 左から行列 $\boldsymbol{Q}$ を掛けることは $\boldsymbol{C}$ の 3 番目の行に 2 を掛けて、最初の行に追加することと同値。Ex3.3 を一般化する。$\boldsymbol{D}_{ij}$ を i 行 j 列目の要素が β であり、それ以外が 0 の行列としたとき、j 番目の行に β を掛けて i 番目の行に追加する $(i \neq j)$ ことは、行列 $\boldsymbol{Q} = \boldsymbol{I} + \boldsymbol{D}_{ij}$ を左に掛けることと同じである。このような行列 $\boldsymbol{Q}$ の行列式は 1 に等しいため、$\boldsymbol{Q}$ は可逆である。

ここで、K 個の行基本変形のシーケンスを考え、その k 番目の演算が特定の可逆行列 $\boldsymbol{Q}_k$ による左乗算に対応すると仮定する。次に、この基本行演算のシーケンスは、可逆行列 $\boldsymbol{Q}_K\boldsymbol{Q}_{K-1}\cdots\boldsymbol{Q}_2\boldsymbol{Q}_1$ による左乗算と同じである。与えられた行列に対して一連の行基本変形を実行することは、その行列に特定の可逆行列を左乗算することと同値であると結論付けることができる。

$\boldsymbol{B}^{-1}\boldsymbol{B} = \boldsymbol{I}$ より、$\boldsymbol{B}^{-1}\boldsymbol{A}_{B(i)}$ は i 番目の単位ベクトル $\boldsymbol{e}_i$ であることが分かる。このことから、$\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$ とすると、

$$\boldsymbol{B}^{-1}\bar{\boldsymbol{B}} = \begin{bmatrix} | & & | & | & | & & | \\ \mathbf{e}_1 & \cdots & \mathrm{e}_{\ell-1} & \mathbf{u} & \mathbf{e}_{\ell+1} & \cdots & \mathbf{e}_m \\ | & & | & | & | & & | \end{bmatrix} = \begin{bmatrix} 1 & & u_1 & & \\ & \ddots & \vdots & & \\ & & u_\ell & & \\ & & \vdots & \ddots & \\ & & u_m & & 1 \end{bmatrix} \tag{5}$$

上記の行列を単位行列に変更する行基本変形を考える。 次の行基本変形のシーケンスを検討する。

(a) それぞれ $i \neq j$ について、l 番目の行を $-\frac{u_i}{u_l}$ 倍したものを i 番目の列に足す。これは、u_i を 0 で置き換えている。

(b) l 番目の行を u_l で割る。これは u_l を 1 で置き換えている。

この行基本変形のシーケンスは、$\boldsymbol{B}^{-1}\bar{\boldsymbol{B}}$ に特定の可逆行列 $\boldsymbol{Q}$ を左乗算することと同じ。結果は同一であるため、$\boldsymbol{QB}^{-1}\bar{\boldsymbol{B}} = \boldsymbol{I}$ となり、$\boldsymbol{QB}^{-1} = \bar{\boldsymbol{B}}^{-1}$ が得られる。つまり、同じ一連の行基本変形を行列 $\boldsymbol{B}^{-1}$ に適用すると $\bar{\boldsymbol{B}}^{-1}$ が得られることを示している。

Ex3.4

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ -2 & 3 & 1 \\ 4 & -3 & -2 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} -4 \\ 2 \\ 2 \end{bmatrix} \tag{6}$$

とし、$l = 3$ と仮定する。つまり、目的はベクトル $\boldsymbol{u}$ を単位ベクトル $\boldsymbol{e}_3 = (0, 0, 1)$ に変換することである。

3 番目の行に 2 を掛けて、最初の行に追加、2 番目の行から 3 番目の行を引く、最後に、3 行目を 2 で割る。そうすると、

$$\bar{\mathbf{B}}^{-1} = \begin{bmatrix} 9 & -4 & -1 \\ -6 & 6 & 3 \\ 2 & -1.5 & -1 \end{bmatrix} \tag{7}$$

が得られる。

説明した方法で $\boldsymbol{B}^{-1}$ を更新することができるため、行列 $\boldsymbol{B}^{-1}$ は各反復の開始時にすでに求められている状態でのシンプレクス法の実装が可能であることが分かる。

修正シンプレクス法の反復

1 通常の反復では、基底列 $\boldsymbol{A}_{B(1)} \cdots \boldsymbol{A}_{B(m)}$、そこから得られる基本実行可能解 $\boldsymbol{x}$、基底行列の逆行列である $\boldsymbol{B}^{-1}$ から開始する。

2 行ベクトル $\boldsymbol{p}' = \boldsymbol{c}'_B \boldsymbol{B}^{-1}$ を計算し、被約費用 $\bar{c}_j = c_j - \boldsymbol{p}' \boldsymbol{A}_j$ を計算する。それらがすべて非負である場合、現在の基本実行可能解が最適であり、アルゴリズムは終了する。それ以外の場合は、$\bar{c}_j < 0$ を満たす j を選択する。

3 $\boldsymbol{u} = \boldsymbol{B}^{-1} \boldsymbol{A}_j$ を計算する。$\boldsymbol{u}$ の成分が正でない場合、最適コストは $-\infty$ であり、アルゴリズムは終了する。

4 $\boldsymbol{u}$ の一部の成分が正の時、$\theta^* = \min_{\{i=1,\ldots,m | u_i > 0\}} \frac{x_{B(i)}}{u_i}$ とする。

5 l を $\theta^* = \frac{x_{B(l)}}{u_l}$ とおく、$\boldsymbol{A}_{B(l)}$ を $\boldsymbol{A}_j$ で置き換えた基底を構成。$y_j = \theta^*, y_{B(i)} = x_{B(i)} - \theta^* u_i (i \neq l)$ である $\boldsymbol{y}$ を新しい基本実行可能解とする。

6 $(m \times (m+1))$ 行列である、$[\boldsymbol{B}^{-1} \mid \boldsymbol{u}]$ を構成。最後の列を単位ベクトル $\boldsymbol{e}_l$ に等しくなるように、各行に l 番目の行の倍数を追加する。最初の m 列が行列 $\bar{\boldsymbol{B}}^{-1}$。

The full tableau implementation

次に、full tableau の観点からシンプレクス法の実装について説明する。ここでは、行列 $\boldsymbol{B}^{-1}$ の維持や更新をする代わりに、列 $\boldsymbol{B}^{-1}\boldsymbol{b}$ や $\boldsymbol{B}^{-1}\boldsymbol{A}_1, \ldots, \boldsymbol{B}^{-1}\boldsymbol{A}_n$ を用いて、$m \times (n+1)$ 行列 $\boldsymbol{B}^{-1}[\boldsymbol{b} \mid \boldsymbol{A}]$ を維持、更新する。この行列を simplex tableau と呼ぶ。

zeroth column と呼ばれる列 $\boldsymbol{B}^{-1}\boldsymbol{b} = \boldsymbol{x}_B$ であることに注意。列 $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ はタブローの i 番目の列と呼ばれる。新たに基底に加えられる変数に対応する列 $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$ は、ピボット列、l 番目の基本変数が基底から外される場合、タブローの l 番目の行はピボット行と呼ばれます。最後に、ピボット行とピボット列の両方に属する要素は、ピボット要素と呼ばれる。ピボット要素 $u_l > 0$ に注意。($\boldsymbol{u} < 0$ の時アルゴリズムはステップ 3 の終了条件を満たしている。)

タブローの行に含まれる情報は、下記の解釈を前提としている。等式制約は、最初は $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ の形式で与えられる。現在の基底行列 B が与えられると、この等式制約はタブローの情報である、$\boldsymbol{B^{-1}b} = \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}$ とも表せる。言い換えると、タブローの行は等式制約 $\boldsymbol{B^{-1}b} = \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}$ の係数を提供する。

各反復の終わりに、タブロー $\boldsymbol{B}^{-1}[\boldsymbol{b} \mid \boldsymbol{A}]$ の更新、$\bar{\boldsymbol{B}}^{-1}[\boldsymbol{b} \mid \boldsymbol{A}]$ の計算をする必要がある。これは、シンプレックスタブローに $\boldsymbol{Q}\boldsymbol{B}^{-l} = \bar{\boldsymbol{B}}^{-1}$ を満たす行列 $\boldsymbol{Q}$ を左乗算することで実現できる。説明したように、これは $\boldsymbol{B}^{-l}$ を $\bar{\boldsymbol{B}}^{-1}$ に変換する基本行演算を実行するのと同じ。つまり、各行にピボット行の倍数を追加し、1 であるピボット要素以外全てのピボット列を 0 にする。

シンプレックス法の要約のステップ 4 と 5、つまり、終了列 $\boldsymbol{A}_{B(l)}$ とステップサイズ θ^* の決定に関して、$\frac{x_{B(i)}}{u_i}$ は、0 番目の列の i 番目のエントリとタブローのピボット列の i 番目のエントリの比率。u_i が正である i のみを考慮し、最小の比率は θ^* に等しく、l が決定できる。

0 番目の行と呼ばれる一番上の行を含めることによってシンプレックスタブローを拡張するのが通例。左上隅のエントリには、現在のコストの負の値である $-\boldsymbol{c}'_B\boldsymbol{x}_B$ が含まれている。(マイナス記号の理由は、単純な更新ルールが可能であるため。) 0 番目の行の残りの部分は、被約費用の行ベクトルです。つまり、ベクトル $\bar{\boldsymbol{c}}' = \boldsymbol{c}' - \boldsymbol{C}'_B\boldsymbol{B}^{-1}\boldsymbol{A}$。したがって、タブローの構造は、

$-\mathbf{c}'_B\mathrm{B}^{-1}\mathbf{b}$	$\mathbf{c}' - \mathbf{c}'_B\mathbf{B}^{-1}$ A
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$

(8)

$-\mathbf{c}'_B\mathbf{x}_B$	$\bar{c}_1$	$\cdots$	$\bar{c}_n$
$x_{B(1)}$	|		|
$\vdots$	$\mathbf{B}^{-1}\mathbf{A}_1$	$\cdots$	$\mathbf{B}^{-1}\mathbf{A}_n$
$x_{B(m)}$	|		|

(9)

となる。

0 番目の行を更新するためのルールは、タブローの他の行に使用されるルールと同じ。ピボット行の倍数を 0 番目の行に追加して、入力変数の被約費用をゼロに設定する。ここで、この更新ルールが 0 番目の行に対しても正しい結果を生成することを確認する。典型的な反復の開始時に、0 番目の行は $[0 \mid \boldsymbol{c}'] - \boldsymbol{g}'[\boldsymbol{b} \mid \boldsymbol{A}](\boldsymbol{g}'[\boldsymbol{b} \mid \boldsymbol{A}])$ という形式に

なる。したがって、0 番目の行は $[0 \mid \boldsymbol{c}']$ と $[\boldsymbol{b} \mid \boldsymbol{A}]$ の行の線形結合。列 j をピボット列、行 l をピボット行とする。ピボット行の形式は $\boldsymbol{h}'[\boldsymbol{b} \mid \boldsymbol{A}]$。ここで、ベクトル $\boldsymbol{h}'$ は $\boldsymbol{B}^{-1}$ の l 番目の行。したがって、ピボット行の倍数が 0 番目の行に追加された後、その行は再び $[0 \mid \boldsymbol{c}']$ と $[\boldsymbol{b} \mid \boldsymbol{A}]$ の（異なる）線形結合であり、あるベクトル $\boldsymbol{p}$ を用いて、$[0 \mid \boldsymbol{c}'] - \boldsymbol{p}'[\boldsymbol{b} \mid \boldsymbol{A}]$ で表される。更新ルールでは、0 番目の行のピボット列エントリがゼロになる。つまり、

$$c_{\bar{B}(\ell)} - \mathbf{p}'\mathbf{A}_{\bar{B}(\ell)} = c_j - \mathbf{p}'\mathbf{A}_j = 0 \tag{10}$$

$i \neq l$ の $\bar{\boldsymbol{B}}_{(i)}$ 番目の列について考える。(これは、基底にとどまる基本変数に対応する列) 基底変数の被約費用のため、基底が変更される前は、その列の 0 番目の行エントリはゼロ。$\bar{\boldsymbol{B}}^{-1}\boldsymbol{A}_{\boldsymbol{B}(\boldsymbol{i})}(i \neq l)$ は i 番目の単位ベクトルであるため、その列のピボット行のエントリもゼロに等しくなる。したがって、タブローの 0 番目の行にピボット行の倍数を追加しても、その列の 0 番目の行エントリには影響せず、0 のまま。新しい基底のすべての列 $\boldsymbol{A}_{\bar{B}(i)}$ について、ベクトル $\boldsymbol{p}$ が $c_{\bar{B}(i)} - \boldsymbol{p}'\boldsymbol{A}_{\bar{B}(i)} = 0$ を満たすと結論付ける。これは、$\boldsymbol{c}'_{\bar{B}} - \boldsymbol{p}'\bar{\boldsymbol{B}} = 0$ や $\boldsymbol{p}' = \boldsymbol{c}'_{\bar{B}}\bar{\boldsymbol{B}}^{-1}$ を意味する。つまり、更新ルールでは、タブローの更新された 0 番目の行は、$[0 \mid \boldsymbol{c}'] - \boldsymbol{c}'_{\bar{B}}\bar{\boldsymbol{B}}^{-1}[\boldsymbol{b} \mid \boldsymbol{A}]$ に等しくなります。

フルタブロー実装の反復

1 通常の反復は、基底行列 $\boldsymbol{B}$ と対応する基本実行可能解 $\boldsymbol{x}$ に関連付けられたタブローから開始する。
2 タブローの 0 行目での被約費用を調べる。それらが全て非負の場合、現在の基本実行可能解が最適であり、アルゴリズムは終了する。それ以外の場合は、$\bar{c}_j < 0$ を満たす j を選択する。
3 タブローの j 番目の列（ピボット列）であるベクトル $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$ を考える。$\boldsymbol{u}$ の成分が正でない場合、最適コストは $-\infty$ であり、アルゴリズムは終了する。
4 u_i が正である各 i について、比率 $\frac{x_{B(i)}}{u_i}$ を計算する。最小の比率に対応する行のインデックスを l とします。列 $A_{B(l)}$ は基底から外れ、列 $\boldsymbol{A}_j$ が基底に加わる
5 タブローの各行に、l 番目の行 (ピボット行) の定数倍を追加して、u_l(ピボット要素) を 1、ピボット列の他のすべてのエントリを 0 にする。

Ex3.5

$$\begin{aligned} \text{minimize} \quad & -10x_1 - 12x_2 - 12x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 \leq 20 \\ & 2x_1 + x_2 + 2x_3 \leq 20 \\ & 2x_1 + 2x_2 + x_3 \leq 20 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{11}$$

Fig3.4 に実行可能範囲が示されている。これを標準形になおすと、

$$\begin{array}{lrcrcrcrcrcl} \text{minimize} & -10x_1 & - & 12x_2 & - & 12x_3 & & & & & & \\ \text{subject to} & x_1 & + & 2x_2 & + & 2x_3 & + & x_4 & & & & = 20 \\ & 2x_1 & + & x_2 & + & 2x_3 & & & + & x_5 & & = 20 \\ & 2x_1 & + & 2x_2 & + & x_3 & & & & & + \quad x_6 & = 20 \\ & & & x_1, & \ldots, & x_6 & \geq 0 & & & & & \end{array} \tag{12}$$

この問題の基本実行可能解は $\boldsymbol{x} = (0, 0, 0, 20, 20, 20)$ であり、ここからアルゴリズムを始める。$B(1) = 4, B(2) = 5, B(3) = 6$ であり、基底行列は単位行列 $\boldsymbol{I}$ である。$\boldsymbol{c}_B = 0, \boldsymbol{c}'_B\boldsymbol{x}_B = 0, \bar{\boldsymbol{c}} = \boldsymbol{c}$ であるため、最初のタブローは、

		x_1	x_2	x_3	x_4	x_5	x_6
	0	-10	-12	-12	0	0	0
$x_4 =$	20	1	2	2	1	0	0
$x_5 =$	20	2*	1	2	0	1	0
$x_6 =$	20	2	2	1	0	0	1

(13)

上記のタブローの形式での注意事項。i 番目の列の上部にあるラベル x_i は、その列に関連する変数を示す。タブローの左側にあるラベル $x_i =$ は、基本変数とその順序を示す。例えば、最初の基本変数 $x_{B(1)} = x_4$ であり、その値は 20。最初の基本変数に関連するタブローの列は、最初の単位ベクトルでなければならない。変数 x_4 に関連する列が最初の単位ベクトルであることがわかるため、x_4 が最初の基本変数となる。x_1 の被約費用は負であり、その変数を基底に加える。ピボット列は $\boldsymbol{u} = (1, 2, 2)$。比率 $\frac{x_{B(i)}}{u_i}(i = 1, 2, 3)$ のうち、最小の比率は、$i = 2, 3$ の時。$l = 2$ とす

る。これにより、アスタリスクで示すピボット要素が決まる。2 番目の基本変数 $x_{B(2)} = x_5$ を基底から外す。新しく加わる基底は $B(1)$ 新たな基底として $\bar{B}(1) = 4, \bar{B}(2) = 1, \bar{3}(1) = 6$ が得られる。

ピボット行に 5 を掛けて、0 番目の行に足す。ピボット行に $frac12$ を掛けて、最初の行から引く。ピボット行を 3 番目から引く。最後に、ピボット行を 2 で割る。これにより、得られたタブローが下である。

		x_1	x_2	x_3	x_4	x_5	x_6
	100	0	−7	−2	0	5	0
$x_4 =$	10	0	1.5	1*	1	−0.5	0
$x_1 =$	10	1	0.5	1	0	0.5	0
$x_6 =$	0	0	1	−1	0	−1	1

(14)

この基本実行可能解は $\boldsymbol{x} = (10, 0, 0, 10, 0, 0)$。Fig3.4 で見ると、点 A から点 D に移動している。$x_6 = 0$ のため、縮退していることに注意。

タブローの行 (0 番目の行を除) は、元の制約 $\boldsymbol{Ax} = \boldsymbol{b}$ と同等の等式制約 $\boldsymbol{B}^{-1}\boldsymbol{Ax} = \boldsymbol{B}^{-l}\boldsymbol{b}$ の表現になる。現在のタブローでは、等式制約を

$$\begin{array}{rcccccccccc} 10 & = & & & 1.5x_2 & + & x_3 & + & x_4 & - & 0.5x_5 \\ 10 & = & x_1 & + & 0.5x_2 & + & x_3 & & & + & 0.5x_5 \\ 1 & = & & & x_2 & - & x_3 & & & - & x_5 \quad + \quad x_6 \end{array} \tag{15}$$

のように表せる。

シンプレックス法に戻ります。 現在のタブローでは、変数 x_2 と x_3 の被約費用がマイナスになっている。次に基底に加えるものとして x_3 を選択。ピボット列は $\boldsymbol{u} = (1, 1, -1)$。$u_3 < 0$ なので、比率 $\frac{x_{B(i)}}{u_i} (i = 1, 2)$ のみを考える。$l = 1$ とし、最初の基本変数 x_4 が基底から外す。ピボット要素もアスタリスクで示されている。先ほどと同様の基本行変形を実行した後、次の新しいタブローを得る。

		x_1	x_2	x_3	x_4	x_5	x_6
	120	0	−4	0	2	4	0
$x_3 =$	10	0	1.5	1	1	−0.5	0
$x_1 =$	0	1	−1	0	−1	1	0
$x_6 =$	10	2.5*	0	1	0	−1.5	1

(16)

Fig3.4 で見ると、点 D から点 B に移動している。また、コストは-120。ここで x_2 のみが負の被約費用を持つ変数のため、x_2 を基底に加え、x_6 を基底から外す。どうようの操作を行うと新たなタブローは、

		x_1	x_2	x_3	x_4	x_5	x_6
	136	0	0	0	3.6	1.6	1.6
$x_3 =$	4	0	0	1	0.4	0.4	−0.6
$x_1 =$	4	1	0	0	−0.6	0.4	0.4
$x_2 =$	4	0	1	0	0.4	−0.6	0.4

(17)

Fig3.4 で見ると、点 B から点 E に移動している。これが最適解であることは、すべての被約費用が負ではないことから確認できる。この例では、シンプレックス法は最適解に到達するために 3 つの基底変換を行い、Fig3.4 の A,D,B,E と調べていった。ピボットルールが異なるとすると、異なるルートで調べることとなる。シンプレックス法は、2 つのエッジ、2 回の反復のみが含まれるルート A,D,E と調べることで最適解を出せたのだろうか。答えはいいえ。最初と最後の基底は 3 つの列で異なるため、少なくとも 3 回の基底の変更が必要。特に、A,D,E と調べる場合、ポイント D で基底変換が縮退する。これにより、合計が 3 回の変換が必要になる。

Ex3.6

シンプレックス法が実際に循環してしまう例。次のピボットルールを使用。

(a) 最も負である被約費用 $\bar{c}_j$ を持つ非基本変数を次に基底に加える変数として選択する。

(b) 基底から外すことのできる基本変数のうち、添字が最小の変数を選択する。

そうすると、p104,105 のようにタブローが変化する。

最後のタブローを見ると、最初と同じ基底を持つタブローとなっている。基底変換のたびに、$\theta^* = 0$ であり、同じコストを持っていた。この例では、同様の操作を何度も繰り返すことになり、シンプレクス法を終了できない。

Comparison of the full tableau and the revised simplex methods

問題が次のように変更されたとする。

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}'\boldsymbol{x} + \boldsymbol{0}'\boldsymbol{y} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{I}\boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{x}, \boldsymbol{y} \geq 0 \end{aligned} \tag{18}$$

この問題にシンプレックス法を実装する。ただし、ベクトル $\boldsymbol{y}$ の要素を決して基底にしないこととし、また、ベクトル $\boldsymbol{y}$ が存在しないものとして基底の変更を行う。この問題における被約費用のベクトルは

$$[\mathbf{c}' \mid 0'] - \mathbf{c}'_B\mathbf{B}^{-1}[\ \mathrm{A} \mid \mathrm{I}] = \left[\bar{\mathbf{c}}' \mid -\mathbf{c}'_B\mathbf{B}^{-1}\right] \tag{19}$$

したがって、この問題のシンプレックスタブローは以下の形式を取る。

$-\mathbf{c}'_B\mathbf{B}^{-1}\mathbf{b}$	$\bar{\mathbf{c}}'$	$-\mathbf{c}'_B\mathbf{B}^{-1}$
$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{B}^{-1}\mathbf{A}$	$\mathbf{B}^{-1}$

(20)

特に、上記のタブローでフルタブロー法のアルゴリズム通りに行うことで、逆基底行列 $\boldsymbol{B}^{-1}$ が各反復で利用できる。これにより、改訂されたシンプレックス法は、タブローの B-1A などの部分が明示的に形成されない代わりに、入力変数 x が選択されると、ピボット列 $\boldsymbol{B}^{-1}\boldsymbol{A}$ をその場で計算すること以外は、上記の拡張問題に適用されたフルタブロー法と本質的に同じであると考えることができる。したがって、改訂されたシンプレックス法は、より効率的な簿記を備えたフルタブロー法に変形したものであることが分かる。改訂されたシンプレックス法が $\boldsymbol{B}^{-1}$ の上にある 0 番目の行エントリも（通常の基本操作によって）更新する場合、シンプレックス乗数 $\boldsymbol{p}' = \boldsymbol{c}'_B\boldsymbol{B}^{-1}$ が利用可能になるため、各反復で線形システム $\boldsymbol{p}'\boldsymbol{B} = \boldsymbol{c}'_B$ を解く必要がなくなる。

ここで、この 2 つの方法の相対的なメリットについて説明する。フルタブロー法では、タブローの各エントリを更新するために、ある一定かつ少数の算術演算が必要となる。したがって、反復ごとの計算量は、タブローのサイズに比例するため、$\mathcal{O}(mn)$。改訂されたシンプレックス法は、同様の計算を使用して $\boldsymbol{B}^{-1}$ と $\boldsymbol{c}'_B\boldsymbol{B}^{-1}$ の更新、つまり $\mathcal{O}(m^2)$ エントリのみが更新されるため、反復ごとの計算量は $\mathcal{O}(m^2)$。さらに、各変数の被約費用は内積 $\boldsymbol{p}'\boldsymbol{A}_j$ を計算することによって求められるため、計算量は $\mathcal{O}(m)$ 最悪の場合、すべての変数について被約費用が計算されるため、反復ごとに合計 $\mathcal{O}(mn)$ の計算量が必要。$m < n$ であるため、どちらの実装でも、反復ごとの最悪の計算量は $\mathcal{O}(mn + m^2) = \mathcal{O}(mn)$。一方、負の被約費用が見つかるまで一つずつの被約費用を計算するピボットルールを検討する場合、改訂されたシンプレックス法の一般的な反復では、はるかに少ない計算量で済む。最良の場合、計算された最初の被約費用が負であり、対応する変数が次に基底に加える変数として選択された場合、合計の計算量は $\mathcal{O}(m^2)$。結論として、改訂されたシンプレックス法はフルタブロー法より計算量が大きくなることはなく、ほとんどの反復ではるかに小さい計算量で計算可能である可能性がある。

改訂されたシンプレクス法のもう一つのメリットは、格納に必要なメモリが $\mathcal{O}(mn)$ から $\mathcal{O}(m^2)$ に削減されていることである。n は m よりはるかに大きいことが多いので、この効果はかなり大きい。行列 $\boldsymbol{A}$ を保存する必要があるため、修正シンプレックス法のメモリ要件も $\mathcal{O}(mn)$ であると考えることもできる。しかし、実際のアプリケーションで発生するほとんどの大規模問題では、行列 $\boldsymbol{A}$ は非常にスパース (0 エントリが多い状態) であり、コンパクトに格納することができる。(なお、$\boldsymbol{A}$ と $\boldsymbol{B}$ がスパースであっても、$\boldsymbol{B}^{-1}\boldsymbol{A}$ は一般にスパースではないので、$\boldsymbol{A}$ のスパース性はフルシンプレックスタブローの格納要領の減少には通常役立たない)。

これまでの議論を以下の表にまとめる。

	Full tableau	Revised simplex
Memory	$O(mn)$	$O\left(m^2\right)$
Worst-case time	$O(mn)$	$O(mn)$
Best-case time	$O(mn)$	$O\left(m^2\right)$

(21)

Practical performance enhancements

中・大規模の問題を解くことを目的としたシンプレックス法の実用的な実装には、線形代数からの多くの考え方が取り入れられているため、簡単に紹介する。

最初の考え方は、再変換に関するものである。修正シンプレクス法の各反復において、逆基底行列 $\boldsymbol{B}^{-l}$ はある規則に従って更新される。このような反復は丸め誤差や切り捨て誤差を生じさせ、それが累積して最終的に非常に不正確な結果をもたらす可能性がある。このような理由から、何回かに一度、$\boldsymbol{B}^{-l}$ を一から計算し直すことが一般的である。このような再計算の効率は、適切なデータ構造と線形代数のある種のテクニックを用いることで大きく向上させることができる。

もう一つの考え方は、逆基底行列 $\bar{\boldsymbol{B}}^{-l}$ の表現方法に関するものである。再変換が行われたばかりで、$\boldsymbol{B}^{-l}$ が利用可能であるとする。修正シンプレックス法の現在の反復の後に、新しい逆基底行列 $\bar{\boldsymbol{B}}^{-l}$ を明示的に生成して保存するオプションがある。$\boldsymbol{Q}\boldsymbol{B}^{-1}$ となるような行列 $\boldsymbol{Q}$ を格納することもできる。これは完全な行列ではなく、m 個の係数で完全に指定することができます。

ここで、$\boldsymbol{u}$ について、$\bar{\boldsymbol{B}}\boldsymbol{u} = \boldsymbol{A}_j$ のシステムを解きたいとする。$\boldsymbol{A}_j$ は、修正シンプレックス法で要求されるように、入力列である。$\boldsymbol{u} = \bar{\boldsymbol{B}}^{-1}\boldsymbol{A}_j = \boldsymbol{Q}\boldsymbol{B}^{-1}\boldsymbol{A}_j$ となり、これは、まず $\boldsymbol{B}^{-1}\boldsymbol{A}_j$ を計算し、次に $\boldsymbol{Q}$ で左乗算（等価に、一連の elelementary row operation を適用）して $\boldsymbol{u}$ を生成できることを示しています。同じ考えで、数回のシンプレックス反復後の逆基底行列を、最初の逆基底行列と $\boldsymbol{Q}$ のようないくつかの疎行列の積として表すことも可能です。

最後に述べるのは、次のようなアイデアである。通常 $\boldsymbol{B}^{-1}$ を明示的に計算するのではなく、$\boldsymbol{B}^{-1}$ を特殊な構造を持つ疎な三角行列の項で表現する。本節で述べる方法は、数値安定性の向上（丸め誤差の影響の最小化）と、問題データのスパース性を利用して実行時間と必要メモリを改善する、という 2 つの目的を達成するために設計されたものである。これらの方法は、実際には決定的な効果をもたらす。数値的に信頼に足る結果を得る可能性が高いだけでなく、シンプレックス法の実行時間を大幅に短縮することができる。これらの手法は、最適化ではなく線形代数のテーマに近いものであるため、これ以上深く説明しない。