行列ゲームのナッシュ均衡

山　下　章　夫

キーワード
ナッシュ均衡　線形相補計画問題　非線形計画問題　*Mathematica*

1 はじめに

本稿は，行列ゲームの混合戦略ナッシュ均衡を求める方法を考察したものである。行列ゲームの混合戦略ナッシュ均衡を求めるための方法は幾つか提示されている。このうち本稿では，Lemke-Howson(5)による方法とMcKelvey(6)による方法を考察する。

第2節では，Lemke-Howsonによる方法を考察する。この方法は，ナッシュ均衡が線形相補計画問題の解に対応することから，その問題を基底の入れ替えによって解くことでナッシュ均衡を求めるものである。同じことだが，線形相補計画問題は相補性条件を考慮に入れると，相補対を変数とする連立方程式を解く問題とみることができる。それゆえ，この連立方程式の非負解を求めることによってナッシュ均衡を求めることができる。

第3節では，McKelveyによる方法を考察する。これはナッシュ均衡が非線形計画問題の解となることを利用している。この節では，最初に2人ゲームで，かつプレーヤーのもつ純粋戦略が2の場合のゲームを考察している。またこの方法は，n人ゲームへの拡張が容易である。それゆえ3人ゲームで，かつ純粋戦略が2の場合のゲームへの適用を試みている。最後に，第4節では，残された課題を述べる。本稿ではナッシュ均衡を求めるためにMathematica[1)]を利用する。また本文中の幾つかの行列ゲームに対応するプログラムを付録に掲載している。

2 線形相補計画問題とナッシュ均衡

この節では，プレーヤーは2人とし，各プレーヤーをプレーヤーⅠとプレーヤーⅡとする。プレーヤーⅠはm個の純粋戦略を持ち，プレーヤーⅡはn個の純粋戦略を持つものとする。各プレーヤーの利得行列を，それぞれA, Bとする。各プレーヤーの混合戦略の集合を

$$X=\{x\in R^m \mid \sum_{i=1}^{m} x_i = 1,\ x_i \geq 0,\ i=1,\ \ldots,\ m\} \qquad (1)$$

$$Y=\{y\in R^n \mid \sum_{j=1}^{n} y_j = 1,\ y_j \geq 0,\ j=1,\ \ldots,\ n\} \qquad (2)$$

で表す。所与の混合戦略$x^*\in X$, $y^*\in Y$に対して，プレーヤーⅠ，Ⅱの期待利得は，それぞれ

1) Version 5.0を利用している。

$(x^*)^TAy^*$, $(x^*)^TBy^*$ である[2)]。

このとき，混合戦略ナッシュ均衡は，すべての $x \in X$，および，すべての $y \in Y$ に対して，不等式

$$(x^*)^TAy^* \geq x^TAy^* \tag{3}$$
$$(x^*)^TBy^* \geq (x^*)^TBy \tag{4}$$

を満たす組，(x^*, y^*) として定義される。(3)はプレーヤーⅡの混合戦略 y^* を所与として，x^* が x^TAy^* を最大化していること，(4)はプレーヤーⅠの混合戦略 x^* を所与として，y^* が $(x^*)^TBy$ を最大化していることを意味する。

すなわち，(3)において，x^* は次の線形計画問題の解である。この線形計画問題を問題 $P1$ とする。

$$\begin{aligned}&(P1)\\&\text{Maximize } x^TAy^* = (Ay^*)^Tx\\&\text{Subject to}\\&Ex = 1\\&x \geq 0\end{aligned} \tag{5}$$

同様に，

(4)において，y^* は次の線形計画問題の解である。そして，この線形計画問題を問題 $P2$ とする。

$$\begin{aligned}&(P2)\\&\text{Maximize } (x^*)^TBy = (B^Tx^*)^Ty\\&\text{Subject to}\\&Fy = 1\\&y \geq 0\end{aligned} \tag{6}$$

ここに，(5)，(6)の E，F は，すべての要素が1のベクトル $E=(1, \ldots, 1) \in R^{1\times m}$，$F=(1, \ldots, 1) \in R^{1\times n}$ を意味する。

いま，問題 $P1$ と問題 $P2$ の双対問題を考える。問題 $P1$ の双対問題を問題 $D1$ とし，双対変数を u とする。そして，問題 $P2$ の双対問題を問題 $D2$ とし，双対変数を v とする。問題 $D1$ と問題 $D2$ は，それぞれ，

$$\begin{aligned}&(D1)\\&\text{Minimize } u\\&\text{Subject to}\\&E^Tu \geq Ay^*\\&u = \text{free}\end{aligned}$$

$$\begin{aligned}&(D2)\\&\text{Minimize } v\\&\text{Subject to}\\&F^Tv \geq B^Tx\\&v = \text{free}\end{aligned}$$

2)　上付き添字の T は転置を意味する。

である。u = free, v = free は u, v には非負条件が課されないことを意味する。

問題 $P1$ の最適解を x^* とし，それの双対問題 $D1$ の最適解を u^* とする。そのとき，双対定理[3]より，$u^*=(x^*)^TAy^*$ を得る。この等式は，(5)より，$u^*Ex^*=u^*$，すなわち，$(x^*)^TE^Tu^*=u^*$ となるので，$(x^*)^T(E^Tu^*-Ay^*)=0$ と表すことができる。これは相補性条件と呼ばれる。同様にして，問題 $P2$ の最適解を y^* とし，それの双対問題 $D1$ の最適解を v^* とする。そのとき，双対定理より，$(y^*)^T(F^Tv^*-B^Tx^*)=0$ を得る。

以上をまとめると，行列ゲーム(A, B)は適当な u, v に対して，次の式が成り立つとき，そしてそのときにのみ，ナッシュ均衡(x, y)を持つ。

$$E^Tu-Ay \geq 0 \tag{7}$$
$$F^Tv-B^Tx \geq 0 \tag{8}$$
$$Ex = 1 \tag{9}$$
$$Fy = 1 \tag{10}$$
$$x^T(E^Tu-Ay)=0 \tag{11}$$
$$y^T(F^Tv-B^Tx)=0 \tag{12}$$
$$x \geq 0,\ y \geq 0 \tag{13}$$

ここで，各プレーヤーの利得行列の要素は非負であるものと仮定する。そして，その行列において列または行のすべての要素がゼロであるような列，行は存在しないものと仮定すると，(7)～(13)は次のようになる。

$$(x')^Tr = 0 \tag{14}$$
$$(y')^Ts = 0 \tag{15}$$
$$Ay'+r = E^T \tag{16}$$
$$B^Tx'+s = F^T \tag{17}$$
$$r,\ s,\ x',\ y' \geq 0 \tag{18}$$
$$v = \frac{1}{Ex'},\ u = \frac{1}{Fy'},\ x = vx',\ y = uy' \tag{19}$$

さらに，(14)，(15)，(16)，(17)は，次のように表すことができる。

$$Mz+w = q \tag{20}$$
$$z^Tw = 0 \tag{21}$$
$$z \geq 0,\ w \geq 0$$

ここに，

$$M=\begin{pmatrix} 0 & A \\ B^T & 0 \end{pmatrix},\ z=\begin{pmatrix} x' \\ y' \end{pmatrix},\ w=\begin{pmatrix} r \\ s \end{pmatrix},\ q=\begin{pmatrix} E^T \\ F^T \end{pmatrix}$$

と置いている。この問題は線形相補計画問題と呼ばれている。z の第 i 番目の要素を z_i，w の第 i 番目の要素を w_i とする。このとき，(z_i, w_i)，$(i=1, \ldots, m+n)$は相補対と呼ばれる。次にその問題を解く一方法である Lemke-Howson 法を考察しよう。

---

3) 双対定理は，問題 $P1$ と問題 $D1$ の一方が最適解をもてば，他方も最適解をもち，問題 $P1$ の目的関数の値と，問題 $D1$ の目的関数の値は一致することを意味する。

掃き出し計算による解

方程式(20)の一つの解は，$z=0$, $w=q$ であることがわかる。Lemke-Howson 法は，基底の入れ替えを行って，相補性条件(21)を満たす $z=0$, $w=q$ 以外の解を求める方法である。z を非基底変数，w を基底変数と呼ぶ。

以下は，掃き出し計算によってナッシュ均衡を求めるアルゴリズムの例である。このアルゴリズムをアルゴリズム1とする。

アルゴリズム1

- 利得行列の要素に負の数がある場合[4]，正数を加えてすべての要素を非負にする。
- M 行列を構成する。
- 最初の基底変数と非基底変数に番号を付ける。基底変数の番号のリストを $aa=\{1, \ldots, m+n\}$，非基底変数の番号のリストを $bb=\{m+n+1, \ldots, 2(m+n)\}$ とする。そして，cc を基底変数と非基底変数の番号の結合リストとする。例えば，$m=2$, $n=2$ の場合，$aa=\{1, 2, 3, 4\}$, $bb=\{5, 6, 7, 8\}$, $cc=\{1, \ldots, 8\}$ である。
- ee を相補対とする。$m=2$, $n=2$ の場合，$ee=\{(1, 5), (2, 6), (3, 7), (4, 8)\}$ である。
- 掃き出し計算の終了判定のために，すべての要素が False である，$m+n$ 個の要素から成るリストをつくる。
- 最初に基底に入れる非基底変数を選び，掃き出し計算を行う。ここでは，基底変数と非基底変数の番号を含めた行列に対して掃き出し計算を行っている。掃き出し計算を行う際に，相補性条件が満たされなければならない。最終表における基底変数の組が，相補性条件を満たす基底変数の組のいずれかになるまで反復計算を行う。ただし，反復計算において基底変数の入れ替えが相補性条件を満たさない状態で繰り返し行われるケースを避けるために，最大の反復回数 qq を設定している。この反復回数を超えると計算を終了する。
- 最終表における基底の組から，第1番目の要素を取り除いたリストを zz とする。そしてそのリストを小さい番号から並べて大きさ2の部分リストをつくり，これを ff とする。
- ff のなかに，相補対が含まれるかどうかを判定し，相補対が含まれる限り基底の入れ替えを行う。
- ナッシュ均衡を表示する。最終表に現れる基底を sol1 とする。kk は最終表における基底変数の番号とその基底変数に対応する値のリストである。ここで，数値は小さいほうから順に並べておく。
- 変数の番号の集合 cc のなかで sol1 に含まれない変数，つまり非基底変数の番号のリストを求める。これらの非基底変数の値はゼロであるから，これらの変数をゼロと置いたリストを rr とする。rr と kk の結合リストを作り，数値の小さい順に並べておく。
- ナッシュ均衡を計算する。
- 最終表における基底変数の組が基底の集合に含まれているかどうか，つまり相補性条件を満たしているかをチェックする。

以上のアルゴリズム1[5]を次の4次の行列に適用した結果は，表1のようになる。

---

4)　ゼロ和2人ゲームの場合，プレーヤーの利得行列に負の要素がある。その場合，これらの負の数のなかで最小の数を求めて，その数の絶対値を利得行列の各要素に加えることによって新たに利得行列を定義する。そしてその利得行列に掃き出し計算を適用する。

5)　アルゴリズム1は付録のプログラム1に対応する。このプログラム1を実行する前に，パッケージ，*DiscreteMath'Combinatorica'* を開いておくことが必要である。この行列は，Judd[4]，p.144を引用。

$$(A,\ B) = \begin{pmatrix} (1,\ 1) & (4,\ 2) & (3,\ 2) & (0,\ 0) \\ (0,\ 3) & (2,\ 2) & (4,\ 1) & (0,\ 10) \\ (0,\ 0) & (5,\ 1) & (3,\ 3) & (10,\ 0) \\ (2,\ 1) & (5,\ 3) & (3,\ 1) & (4,\ 4) \end{pmatrix} \tag{22}$$

表1　掃き出し計算のための表と最終表

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	0	0	0	0	0	0	0	0	0	0	0	1	4	3	0	1
2	0	1	0	0	0	0	0	0	0	0	0	0	0	2	4	0	1
3	0	0	1	0	0	0	0	0	0	0	0	0	0	5	3	10	1
4	0	0	0	1	0	0	0	0	0	0	0	0	2	5	3	4	1
5	0	0	0	0	1	0	0	0	1	3	0	1	0	0	0	0	1
6	0	0	0	0	0	1	0	0	2	2	1	3	0	0	0	0	1
7	0	0	0	0	0	0	1	0	2	1	3	1	0	0	0	0	1
8	0	0	0	0	0	0	0	1	0	10	0	4	0	0	0	0	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	$-\left(\frac{3}{4}\right)$	0	0	0	0	0	0	0	0	0	0	1	$\frac{5}{2}$	0	0	$\frac{1}{4}$
15	0	$\frac{1}{4}$	0	0	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$	1	0	$\frac{1}{4}$
16	0	$-\left(\frac{3}{40}\right)$	$\frac{1}{10}$	0	0	0	0	0	0	0	0	0	0	$\frac{7}{20}$	0	1	$\frac{1}{40}$
4	0	$-\left(\frac{9}{20}\right)$	$-\left(\frac{2}{5}\right)$	1	0	0	0	0	0	0	0	0	2	$\frac{21}{10}$	0	0	$\frac{3}{20}$
5	0	0	0	0	1	0	0	$-\left(\frac{3}{10}\right)$	1	0	0	$-\left(\frac{1}{5}\right)$	0	0	0	0	$\frac{7}{10}$
6	0	0	0	0	0	1	$-\left(\frac{1}{3}\right)$	$-\left(\frac{1}{6}\right)$	$\frac{4}{3}$	0	0	2	0	0	0	0	$\frac{1}{2}$
11	0	0	0	0	0	0	$\frac{1}{3}$	$-\left(\frac{1}{30}\right)$	$\frac{2}{3}$	0	1	$\frac{1}{5}$	0	0	0	0	$\frac{3}{10}$
10	0	0	0	0	0	0	0	$\frac{1}{10}$	0	1	0	$\frac{2}{5}$	0	0	0	0	$\frac{1}{10}$

最終表の第1列の数値は基底変数を示している。この行列ゲームの例では，最初の基底変数の番号は{1, 2, 3, 4, 5, 6, 7, 8}であり，非基底変数の番号は{9, 10, 11, 12, 13, 14, 15, 16}である。相補性条件は，変数の番号のそれぞれの対{(1, 9), (2, 10), (3, 11), (4, 12), (5, 13), (6, 14), (7, 15), (8, 16)}において，どちらかの番号の変数がゼロでなければならないことを意味する。つまり，これらの対のどちらかが最終表の第1列に現れなければならない。表1の最終表の第1列の数値は，この条件を満たしている。リストの要素がすべてFalseであれば相補性条件は満たされることを意味する。リストの要素に一つでもTrueがあれば相補性条件が満たされないことを意味する。この場合は，リストの要素はすべてFalseとなっている。これは，プログラムを実行した後で，

Table [MemberQ[ff, ee[[i]]], {i, 1, m+n}]

を入力すれば，

{False, False, False, False, False, False, False, False}

となることから，相補性条件は満たされていることがわかる。一つのナッシュ均衡は$(x_1, x_2, x_3, x_4)=\left(0, \frac{1}{4}, \frac{3}{4}, 0\right)$, $(y_1, y_2, y_3, y_4)=\left(0, 0, \frac{10}{11}, \frac{1}{11}\right)$である。このプログラムでは，異なる非基底変数から出発して計算を行うと，他のナッシュ均衡に到達する場合がある。また，求められないナッシュ均衡もある。

線形相補方程式の解

以上みたように，Lemke-Howsonによる方法は，掃き出し計算によってナッシュ均衡を求めるものである。線形相補方程式，(20)，(21)において方程式の数は$2(m+n)$本，変数は$2(m+n)$個であり，変数と方程式の数は一致する。したがって，もしこの方程式を直接解いて，非負条件を満たす解を得ることができれば，それらの解に対応するナッシュ均衡を求めることができる。線形相補方程式を直接解いて，ナッシュ均衡を求めるアルゴリズムは以下のとおりである。

アルゴリズム2

- 利得行列の要素を正数にする。
- M行列を構成する。
- 線形相補方程式のすべての解を求める。
- 解のなかで非負条件を満たすものを取り出す。
- ナッシュ均衡を表示する。

以上の方法によるプログラムは，付録のプログラム2[6]である。(22)の行列ゲームに適用すると，次の複数の解を得る。

$$(x_1, x_2, x_3, x_4)=\left(0, 0, \frac{1}{2}, \frac{1}{2}\right), \quad (y_1, y_2, y_3, y_4)=\left(0, \frac{1}{4}, \frac{3}{4}, 0\right)$$

$$(x_1, x_2, x_3, x_4)=\left(0, 0, \frac{1}{2}, \frac{1}{2}\right), \quad (y_1, y_2, y_3, y_4)=(0, 1, 0, 0)$$

$$(x_1, x_2, x_3, x_4)=\left(0, \frac{1}{4}, \frac{3}{4}, 0\right), \quad (y_1, y_2, y_3, y_4)=\left(0, 0, \frac{10}{11}, \frac{1}{11}\right)$$

3　非線形計画問題とナッシュ均衡

この節では，ナッシュ均衡を非線形計画問題の解として求める方法を考察する。最初に2人非ゼロ和ゲームの解を求める問題を非線形計画問題として定式化する。プレーヤーの持つ純粋戦略の数が2の場合の各プレーヤーの最適反応を図解する。そして，3人ゲームへの適用を試みる。

最初に，2人ゲームを非線形計画問題として定式化しよう。ナッシュ均衡の定義式(3)，(4)の不等式の左辺をそれぞれv_1, v_2と置くと，(3)，(4)はそれぞれ

$$v_1 \geq \sum_{j=1}^{n} a_{ij} y_j^*, \quad (i = 1, \ldots, m) \tag{23}$$

$$v_2 \geq \sum_{i=1}^{m} b_{ij} x_i^*, \quad (j = 1, \ldots, n) \tag{24}$$

6)　このプログラムはMathematicaのSolve関数を直接，利用したものである。この行列ゲームでは，Mathematicaは，すべての解を求めることができないというメッセージを表示する。しかし，掃き出し計算のプログラムでは求めることができなかった解を得ることができる。

と表すことができる。$\sum_{j=1}^{n} a_{ij}y_j^*$ はプレーヤーⅠが純粋戦略 i を選び，プレーヤーⅡが混合戦略 $(y_1^*, \ldots, y_n^*)$ を選んだときのプレーヤーⅠの期待利得である。$\sum_{i=1}^{m} b_{ij}x_i^*$ は，プレーヤーⅡが純粋戦略 j を選び，プレーヤーⅠが混合戦略 $(x_1^*, \ldots, x_m^*)$ を選んだときのプレーヤーⅡの期待利得である。

McKelvey によれば，ナッシュ均衡は，次式で定義される関数[7]の最小値に対応する独立変数の値である。

$$f(x^*, y^*) = \left(Max\left(\sum_{j=1}^{n} a_{1j}y_j^* - v_1, 0\right)\right)^2 + \cdots + \left(Max\left(\sum_{j=1}^{n} a_{mj}y_j^* - v_1, 0\right)\right)^2 + \left(Max\left(\sum_{i=1}^{m} b_{i1}x_i^* - v_2, 0\right)\right)^2 + \cdots + \left(Max\left(\sum_{i=1}^{m} b_{in}x_i^* - v_2, 0\right)\right)^2 \quad (25)$$

明らかに，$f(x^*, y^*) \geq 0$ である。また，(23)，(24) が成り立つような (x^*, y^*) に対して，$f(x^*, y^*) = 0$ である。それゆえ，ナッシュ均衡を求める問題は，次の非線形計画の問題となる。

$$\text{Minimize } f(x, y) = \sum_{i=1}^{m}\left(Max\left(\sum_{j=1}^{n} a_{ij}y_j - v_1, 0\right)\right)^2 + \sum_{j=1}^{n}\left(Max\left(\sum_{i=1}^{m} b_{ij}x_i - v_2, 0\right)\right)^2 \quad (26)$$

Subject to

$$x_1 + \cdots + x_m = 1$$

$$y_1 + \cdots + y_n = 1$$

$$x_1, \ldots, x_m, y_1, \ldots, y \geq 0$$

したがって，この方法のアルゴリズムは次のようになる。

アルゴリズム 3

- 目的関数を構成する。
- 目的関数の最小値を求める。
- 最小値がゼロに十分近いかどうかを判定する。ゼロに近ければ，その最小値に対応する独立変数の値を求める。
- ナッシュ均衡を表示する。

付録のプログラム 3 では，次の行列ゲームをペナルティ関数法[8]で解いている。

$$(A, B) = \begin{pmatrix} (1, 5) & (4, 6) \\ (2, 3) & (0, 2) \end{pmatrix} \quad (27)$$

また，$m=2$, $n=2$ のケースでは，プレーヤーの最適反応をグラフに描くことができるので，最適反応のグラフ[9]を描いている。この図から，3 つのナッシュ均衡が存在することがわかる。

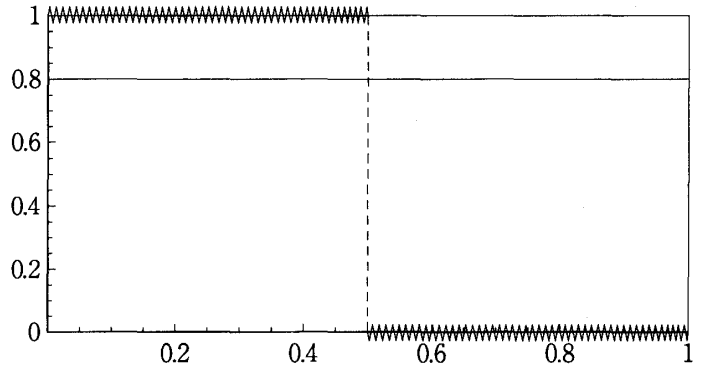
---

7) 関数の記号 $Max[a, b]$ は，$a \geq b$ のとき，$Max[a, b] = a$ を意味する。

8) ペナルティ関数法の理論については，例えば，坂和[2]を参照。

9) この最適反応のグラフを描くためには，プログラム 3 を実行する前に，パッケージ *Graphics'MultipleListPlot'* を開いておくことが必要である。

図1　最適反応のグラフ



McKelveyの方法は，3人ゲームへ適用することができる。3人のプレーヤーを，Ⅰ，Ⅱ，Ⅲとする。そして簡単化のために，各プレーヤーは2つの純粋戦略を持っているものとする。この3人ゲームの利得表を次のように表しておこう。

プレーヤーⅢが純粋戦略1をとるとき，各プレーヤーの利得は

$$Q=\begin{pmatrix}(a_{111},\ b_{111},\ c_{111}) & (a_{121},\ b_{121},\ c_{121})\\ (a_{211},\ b_{211},\ c_{211}) & (a_{221},\ b_{221},\ c_{221})\end{pmatrix} \tag{28}$$

であり，プレーヤーⅢが純粋戦略2をとるとき，各プレーヤーの利得は

$$Q'=\begin{pmatrix}(a_{112},\ b_{112},\ c_{112}) & (a_{122},\ b_{122},\ c_{122})\\ (a_{212},\ b_{212},\ c_{212}) & (a_{222},\ b_{222},\ c_{222})\end{pmatrix} \tag{29}$$

である。この利得表は，例えば，プレーヤーⅠが純粋戦略1，プレーヤーⅡが純粋戦略1，プレーヤーⅢが純粋戦略1をそれぞれとるとき，各プレーヤーの利得は$(a_{111},\ b_{111},\ c_{111})$，またプレーヤーⅠが純粋戦略2，プレーヤーⅡが純粋戦略1，プレーヤーⅢが純粋戦略2をとるとき，各プレーヤーの利得は$(a_{212},\ b_{212},\ c_{212})$であることを意味する。

いま，3人のプレーヤー，Ⅰ，Ⅱ，Ⅲの混合戦略の集合を，それぞれ

$$X=\{x\in R^2\,|\,x_1+x_2=1,\ x_1\geq 0,\ x_2\geq 0\} \tag{30}$$
$$Y=\{y\in R^2\,|\,y_1+y_2=1,\ y_1\geq 0,\ y_2\geq 0\} \tag{31}$$
$$Z=\{z\in R^2\,|\,z_1+z_2=1,\ z_1\geq 0,\ z_2\geq 0\} \tag{32}$$

とする。

すべてのプレーヤーが混合戦略をとるとき，プレーヤーⅠ，Ⅱ，Ⅲの期待利得をそれぞれ$f(x,\ y,\ z)$，$g(x,\ y,\ z)$，$h(x,\ y,\ z)$とすると，それらは次式で表される。

$$f(x,\ y,\ z)=\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2}x_iy_jz_ka_{ijk} \tag{33}$$

$$g(x,\ y,\ z)=\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2}x_iy_jz_kb_{ijk} \tag{34}$$

$$h(x,\ y,\ z)=\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2}x_iy_jz_kc_{ijk} \tag{35}$$

このとき，混合戦略ナッシュ均衡は，すべての$x\in X$，$y\in Y$，$z\in Z$に対して，不等式

$$\alpha \equiv f(x^*, y^*, z^*) \geq f(x, y^*, z^*) \tag{36}$$
$$\beta \equiv g(x^*, y^*, z^*) \geq g(x^*, y, z^*) \tag{37}$$
$$\gamma \equiv h(x^*, y^*, z^*) \geq h(x^*, y^*, z) \tag{38}$$

を満たす組，(x^*, y^*, z^*)として定義される。

ナッシュ均衡の定義式(36)，(37)，(38)は次式を意味する[10]。

$$\alpha \geq \sum_{j=1}^{2}\sum_{k=1}^{2} y_j z_k a_{1jk} \tag{39}$$
$$\alpha \geq \sum_{j=1}^{2}\sum_{k=1}^{2} y_j z_k a_{2jk} \tag{40}$$
$$\beta \geq \sum_{i=1}^{2}\sum_{k=1}^{2} x_i z_k b_{i1k} \tag{41}$$
$$\beta \geq \sum_{i=1}^{2}\sum_{k=1}^{2} x_i z_k b_{i2k} \tag{42}$$
$$\gamma \geq \sum_{i=1}^{2}\sum_{j=1}^{2} x_i y_j c_{ij1} \tag{43}$$
$$\gamma \geq \sum_{i=1}^{2}\sum_{j=1}^{2} x_i y_j c_{ij2} \tag{44}$$

(39)の右辺は，プレーヤー1が純粋戦略1をとり，他のプレーヤーが混合戦略をとるときのプレーヤー1の期待利得である。(40)の右辺はプレーヤー1が純粋戦略2をとり，他のプレーヤーが混合戦略をとるときのプレーヤー1の期待利得である。同様に，(41)～(44)の右辺は当該プレーヤーが純粋戦略1，2をとり，他のプレーヤーが混合戦略をとるときの当該プレーヤーの期待利得である。

このとき，ナッシュ均衡を求める問題は，次の関数$\phi(x, y, z)$の最小値を求める非線形計画問題となる。

$$\begin{aligned}\text{Minimize } \phi(x, y, z)\\ =&\left(\text{Max}\left(\sum_{j=1}^{2}\sum_{k=1}^{2} y_j z_k a_{1jk} - \alpha,\ 0\right)\right)^2 + \left(\text{Max}\left(\sum_{j=1}^{2}\sum_{k=1}^{2} y_j z_k a_{2jk} - \alpha,\ 0\right)\right)^2\\ &+\left(\text{Max}\left(\sum_{i=1}^{2}\sum_{k=1}^{2} x_i z_k b_{i1k} - \beta,\ 0\right)\right)^2 + \left(\text{Max}\left(\sum_{i=1}^{2}\sum_{k=1}^{2} x_i z_k b_{i2k} - \beta,\ 0\right)\right)^2\\ &+\left(\text{Max}\left(\sum_{i=1}^{2}\sum_{j=1}^{2} x_i y_j c_{ij1} - \gamma,\ 0\right)\right)^2 + \left(\text{Max}\left(\sum_{i=1}^{2}\sum_{j=1}^{2} x_i y_j c_{ij2} - \gamma,\ 0\right)\right)^2\end{aligned} \tag{45}$$

Subject to

$$x_1 + x_2 = 1 \tag{46}$$
$$y_1 + y_2 = 1 \tag{47}$$
$$z_1 + z_2 = 1 \tag{48}$$
$$x_i,\ y_i,\ z_i \geq 0,\ (i = 1, 2) \tag{49}$$

(45)の右辺は，(39)～(44)の右辺と左辺の差とゼロを比較して，そのうち大きい方をとり2乗して合計したものである。関数の定義から，$\phi(x, y, z) \geq 0$である。そして，この関数の最小値はゼロであり，その最小値に対応する変数の値がナッシュ均衡である。プログラム4はこの問題にペナ

10) 記号の節約のために，これらの不等式の右辺の変数にある記号＊は省略している。

ルティ関数法を適用してナッシュ均衡を求めている。付録のプログラム4を実行すると5つのナッシュ均衡を得る。

この行列ゲームの場合，それらの5つのナッシュ均衡の厳密解は次のようにして求めることができる。解のうちの一つの内部解は期待利得の式(33)，(34)，(35)に制約条件(46)，(47)，(48)の関係を代入して3変数(x_1, y_1, z_1)に関する式をつくり，連立方程式

$$\frac{\partial f(x_1, y_1, z_1)}{\partial x_1}=0,\ \frac{\partial g(x_1, y_1, z_1)}{\partial y_1}=0,\ \frac{\partial h(x_1, y_1, z_1)}{\partial z_1}=0, \tag{50}$$

を解くことによって得られる。それは$(x_1, y_1, z_1)=\left(2-\sqrt{2},\ 2-\sqrt{2},\ \frac{10\sqrt{2}-9}{7}\right)$である。

また，この3人ゲームのナッシュ均衡は，連立不等式を解く[11]ことによって求めることができる。(39)～(44)は(x_1, y_1, z_1)に関する連立不等式とみることができる。それゆえ制約条件(46)，(47)，(48)，(49)を考慮に入れて，これらの変数に関する連立不等式を直接解くと，次の5つのナッシュ均衡を得る。

$$(x_1, y_1, z_1)=(0, 0, 0)$$
$$(x_1, y_1, z_1)=\left(\frac{1}{4}, \frac{1}{4}, 0\right)$$
$$(x_1, y_1, z_1)=\left(2-\sqrt{2}, x_1, \frac{1-4x_1}{-3+2x_1}\right)$$
$$(x_1, y_1, z_1)=\left(\frac{2}{3}, \frac{2}{3}, 1\right)$$
$$(x_1, y_1, z_1)=(1, 1, 1)$$

4　おわりに

本稿では，標準形ゲームのナッシュ均衡を求める方法を考察した。ここで考察した方法は，ナッシュ均衡が線形相補計画問題の解となることに着目して，その線形相補計画問題の解を掃き出し計算によって求める方法，線形相補方程式を直接解いて非負解に対応するナッシュ均衡を選び出す方法，非線形計画問題を解いてナッシュ均衡を求める方法である。

展開形ゲームは，本稿で考察したような標準形ゲームへ変換できるが，行列の次数が大きくなる点が指摘されている。展開形ゲームにおいて，効率的にナッシュ均衡を求める方法の研究は今後の課題としたい。

参考文献

[1] 小林道正『*Mathematica* による線形代数』朝倉書店　1996年。
[2] 坂和正敏『非線形システムの最適化』森北出版　1986年。
[3] 船木由喜彦『演習ゲーム理論』新世社　2004年。
[4] Judd K. L., *Numerical Methods in Economics*, The MIT Press, 1998.
[5] Lemke, C. E. and Howson, J. T. Jr.,"Equilibrium points of bimatrix games," *Journal of the Society for Industrial and Applied Mathematics* 12, 1964.

11)　連立不等式を解くとき，Reduce 関数か InequalitySolve 関数を使う。この方法は，3節の(27)のような各プレーヤーの純粋戦略の数が2の行列ゲームには有効である。

[6] McKelvey R. D. and Mclennan A., "Computation of Equilibria in Finite Games," in *Handbook of Computational Economics*, Volume 1, Elsevier Science B. V., 1996.

[7] von Stengel, B., "Computing Equilibria for Two-Person Games," 2002. available from http://www.maths.lse.ac.uk/Personal/stengel/bvs-publ.html.

(受理 2005年12月19日)

付 録

プログラム1

```
m=4 ; n=4 ; (*m×n matrix*) a={{1, 4, 3, 0}, {0, 2 ,4 ,0}, {0, 5, 3, 10}, {2, 5, 3, 4}};
b={{1, 2, 2, 0}, {3, 2, 1, 10}, {0, 1, 3, 0}, {1, 3, 1, 4}};
bT=Transpose [b] ; s=Table [0, {m}] ; nn=Table [0, {n}] ; q=Table [{1}, {m+n}];
mat=Join [Table [Join [s, a [[i]]], {i, 1, m}], Table [Join [bT [[j]], nn], {j, 1, n}]];
d=Table [Join [Identity Matrix [m+n] [[i]], mat [[i]], q [[i]]], {i, 1, m+n}];
num=List [Table [i, {i, 1, 2*(m+n) +1}]];
aa=Table [i, {i, 1, m+n}];
bb=Table [i, {i, m+n+1, 2*(m+n)}];
ee=Table [{aa [[i]], bb [[i]]}, {i, 1, m+n}];
e=Table [{i}, {i, 0, m+n}];
c=Table [Join [e [[i]], Join [num, d] [[i]]], {i, 1, (m+n)+1}];
MatrixForm[c]
df=Table [False, {m+n}];
z1=2 ; (*1≤z1≤m+n*) (qq=1 ; Label [Iteration]; z=z1;
 r=Table [If [c [[i, (m+n+1)+z]]>0,
    c [[i, 2*(m+n) +2]] /c [[i, (m+n+1)+z]], Infinity], {i, 2, m+n+1}];
 z1=Catch [Table [{If [Min [r]=r [[i]], Throw [i]]}, {i, 1, m+n}]];
 p=c [[z1+1]] /c [[z1+1, (m+n+1)+z]];
 v=Table [Table [c [[k, j]]-c [[k, (m+n+1) +z]] *p [[j]], {j, 1, 2*(m+n)+2}],
    {k, 1, m+n+1}];
 f=Delete [Replace Part [v, p, z1+1], 1];
 d1=Table [Delete [f [[i]], 1], {i, 1, m+n}];
 e [[z1+1]] =List [m+n+z];
 c=Table [Join [e [[i]], Join [num, d1] [[i]]], {i, 1, (m+n)+1}]; qq+=1;
 zz=Delete [Flatten [e], 1];
 ff=K Subsets [Sort [zz], 2];
 If [Table [Member Q [ff, ee [[i]]], {i, 1, m+n}]≠df&&qq<100, Goto [Iteration]];)
Table Form [c]
(*Nash Equilibrium*)
cc=Table [i, {i, 1, 2*(m+n)}];
sol1=Table [c [[i, 1]], {i, 2, m+n+1}];
kk=Sort [Table [List [c [[i, 1]], c [[i, 2*(m+n) +2]]], {i, 2, m+n+1}]];
hh=Sort [Complement [cc, Sol1]];
rr=Table [List [hh [[i]], 0], {i, 1, Length [hh]}];
sol2=Sort [Join [rr, kk]];
```

```
xx=Table [sol2 [[(m+n)+j, 2]] /Sum [sol2 [[(m+n)+i, 2]], {i, 1, m}], {j, 1, m}];
yy=Table [sol2 [[(m+n)+j, 2]] /Sum [sol2 [[(m+n)+i, 2]], {i, m+1, m+n}], {j, m+1, m+n}];
List [xx, yy]
```

プログラム2

```
m=4 ; n=4 ; (*m×n matrix*) a={{1, 4, 3, 0}, {0, 2, 4, 0}, {0, 5, 3, 10}, {2, 5, 3, 4}};
b={{1, 2, 2, 0}, {3, 2, 1, 10}, {0, 1, 3, 0}, {1, 3, 1, 4}};
bT=Transpose [b] ; s=Table [0, {m}] ; nn=Table [0, {n}] ; q=Table [{1}, {m+n}];
mat=Join [Table [Join [s, a [[i]]], {i, 1, m}], Table [Join [bT [[j]], nn], {j, 1, n}]];
ww=Table [w [t], {t, 1, m+n}];
zz=Table [z [t], {t, 1, m+n}];
f=Table [1, {m+n}];
eq1=Table [(mat. zz+ww) [[i]]=f [[i]], {i, 1, m+n}];
eq2=Table [(ww. zz) [[i]]=0, {i, 1, m+n}];
(*complementary condition*)sol=Solve [Join [eq1, eq2], Join [ww, zz]];
sol1=Table [Table [If [sol [[i, j, 2]]≥0, True], {j, 1, 2*(m+n)}], {i, 1, Length [sol]}];
df=Table [True, {2*(m+n)}];
sol2=Table [If [sol1 [[i]]=df, i], {i, 1, Length [sol1]}];
ss=Cases [sol2, _Integer];
data=Table [sol [[ss [[i]]]], {i, 1, Length [ss]}];
(*Nash Equilibrium*)
xx [t_] :=Table [data [[t, (m+n)+j, 2]] /Sum [data [[t, (m+n)+i, 2]], {i, 1, m}], {j, 1, m}];
yy [t_] :=Table [data [[t, (m+n)+j, 2]] /Sum [data [[t, (m+n)+i, 2]], {i, m+1, m+n}],
  {j, m+1, m+n}];
sol=Table [If [Sum [data [[t, (m+n)+i, 2]], {i, 1, m}]>0&&
  Sum [data [[t, (m+n)+i, 2]], {i, m+1, m+n}]>0,
  {xx [t], yy [t]},""], {t, 1, Length [data]}]
```

プログラム3

```
<<Graphics 'Multiple List Plot'
x={x1, 1−x1};
y={y1, 1−y1};
a={{1, 4}, {2, 0}};
b={{5, 6}, {3, 2}};
bT=Transpose [b];
h1 [x1_, y1_] :=x. a. y ;
h2 [x1_, y1_] :=x. b. y ;
step=0.01 ;
s=Table [h1 [x1, y1], {y1, 0, 1.0, step}, {x1, 0, 1.0, step}];
z=Table [{Catch [Table [{If [Max [s [[j]]]=s [[j, i]], Throw [i]]}, {i, 1, (1.0−0) /step+1}]]},
  {j, 1, (1.0−0) /step+1}];
opt=Flatten [0+(z−1)*step];
xx=Table [i, {i, 0, 1.0, step}];
graph1=Table [{opt [[i]], xx [[i]]}, {i, 1, (1.0−0) /step+1}];
```

```
u=Table [h2 [x1, y1], {x1, 0, 1.0, step}, {y1, 0, 1.0, step}];
w=Table [{Catch [Table [{If [Max [u [[j]]]=u [[j, i]], Throw [i]]}, {i, 1, (1.0−0) /step+1}]]},
    {j, 1, (1.0−0) /step+1}];
vv=Flatten [0+(w−1)*step];
graph2=Table [{xx [[i]], vv [[i]]}, {i, 1, (1.0−0) /step+1}];
Multiple List Plot [graph1, graph2, Plot Joined→True]
(*McKelvey's Method*)
q=10^4 ;
v [x1__, y1__] :=If [a [[1]]. y−x. a. y≤0, 0, a [[1]]. y−x. a. y]^2+
  If [a [[2]].y−x. a. y≤0, 0, a [[2]]. y−x. a. y]^2+
  If [bT [[1]]. x−x. b. y≤0, 0, bT [[1]]. x−x. b. y]^2+
  If [bT [[2]]. x−x. b. y≤0, 0, bT [[2]]. x−x. b. y]^2+q*(If [−x1≤0, 0, −x1]^2+
  If [−y1≤0, 0, −y1]^2+If [x1−1≤0, 0, x1−1]^2+If [y1−1≤0, 0, y1−1]^2);
tt=0.2 ; (*初期値の変化*)r [x0__, y0__] : =Find Minimum [v [x1, y1],
    {x1, x0, x0+0.1}, {y1, y0, y0+0.1}];
sol=Flatten [Table [r [x0, y0], {x0, 0, 1.0, tt}, {y0, 0, 1.0, tt}], 1];
tol=1.0*10^(−5);
data1=Table [If [sol [[i, 1]]<tol, i, "nonopt"], {i, 1, Length [sol]}];
ss=Cases [data1, _Integer];
Matrix Form [Table [sol [ss [[i]]]], {i, 1, Length [ss]}]]
```

プログラム4

```
(*利得行列の定義*)
a [1, 1, 1]=4 ; a [1, 2, 1]=0 ; a [2, 1, 1]=0 ; a [2, 2, 1]=8 ;
a [1, 1, 2]=6 ; a [1, 2, 2]=0 ; a [2, 1, 2]=0 ; a [2, 2, 2]=2 ;
b [1, 1, 1]=2 ; b [1, 2, 1]=0 ; b [2, 1, 1]=0 ; b [2, 2, 1]=4 ;
b [1, 1, 2]=3 ; b [1, 2, 2]=0 ; b [2, 1, 2]=0 ; b [2, 2, 2]=1 ;
c [1, 1, 1]=−4 ; c [1, 2, 1]=0 ; c [2, 1, 1]=0 ; c [2, 2, 1]=−6 ;
c [1, 1, 2]=−6 ; c [1, 2, 2]=0 ; c [2, 1, 2]=0 ; c [2, 2, 2]=−2 ;
(*期待利得*)
ff=Simplify [Sum [x [i]*y [j]*z [k]*a [i, j, k], {i, 1, 2}, {j, 1, 2}, {k, 1, 2}] /.
    {x [2]→1−x [1], y [2]→1−y [1], z [2]→1−z [1]}];
gg=Simplify [Sum [x [i]*y [j]*z [k]*b [i, j, k], {i, 1, 2}, {j, 1, 2}, {k, 1, 2}] /.
    {x [2]→1−x [1], y [2]→1−y [1], z [2]→1−z [1]}];
hh=Simplify [Sum [x [i]*y [j]*z [k]*c [i, j, k], {i, 1, 2}, {j, 1, 2}, {k, 1, 2}] /.
    {x [2]→1−x [1], y [2]→1−y [1], z [2]→1−z [1]}];
f [x1__, y1__, z1__] :=ff /. {x [1]→x1, y [1]→y1, z [1]→z1};
g [x1__, y1__, z1__] :=gg /. {x [1]→x1, y [1]→y1, z [1]→z1};
h [x1__, y1__, z1__] :=hh /. {x [1]→x1, y [1]→y1, z [1]→z1};
a1 [y1__, z1__] :=Simplify [Sum [y [j]*z [k]*a [1, j, k], {j, 1, 2}, {k, 1, 2}] /.
    {y [2]→1−y [1], z [2]→1−z [1]} /. {y [1]→y1, z [1]→z1}];
```

```
a2 [y1__, z1__] :=Simplify [Sum[Sum[y [j] *z [k] *a [2, j, k], {k, 1, 2}], {j, 1, 2}] /.
    {y [2]→1−y [1], z [2]→1−z [1]}/.{y [1]→y1, z [1]→z1}];
b1 [x1__, z1__] :=Simplify [Sum[Sum[x [i] *z [k] *b [i, 1, k], {k, 1, 2}], {i, 1, 2}] /.
    {x [2]→1−x [1], z [2]→1−z [1]}/.{x [1]→x1, z [1]→z1}];
b2 [x1__, z1__] :=Simplify [Sum[Sum[x [i] *z [k] *b [i, 2, k], {k, 1, 2}], {i, 1, 2}] /.
    {x [2]→1−x [1], z [2]→1−z [1]}/.{x [1]→x1, z [1]→z1}];
c1 [x1__, y1__] :=Simplify [Sum[Sum[x [i] *y [j] *c [i, j, 1], {j, 1, 2}], {i, 1, 2}] /.
    {x [2]→1−x [1], y [2]→1−y [1]}/.{x [1]→x1, y [1]→y1}];
c2 [x1__, y1__] :=Simplify [Sum[Sum[x [i] *y [j] *c [i, j, 2], {j, 1, 2}], {i, 1, 2}] /.
    {x [2]→1−x [1], y [2]→1−y [1]}/.{x [1]→x1, y [1]→y1}];
q=10^4 ;
v [x1__, y1__, z1__] :=If [a1 [y1, z1]−f [x1, y1, z1]≤0, 0, a1 [y1, z1]−f [x1, y1, z1]]^2+
  If [a2 [y1, z1]−f [x1, y1, z1]≤0, 0, a2 [y1, z1]−f [x1, y1, z1]]^2+
  If [b1 [x1, z1]−g [x1, y1, z1]≤0, 0, b1 [x1, z1]−g [x1, y1, z1]]^2+
  If [b2 [x1, z1]−g [x1, y1, z1]≤0, 0, b2 [x1, z1]−g [x1, y1, z1]]^2+
  If [c1 [x1, y1]−h [x1, y1, z1]≤0, 0, c1 [x1, y1]−h [x1, y1, z1]]^2+
  If [c2 [x1, y1]−h [x1, y1, z1]≤0, 0, c2 [x1, y1]−h [x1, y1, z1]]^2+
  q* (If [−x1≤0, 0, −x1]^2+If [−y1≤0, 0, −y1] ^2+If [−z1≤0, 0, −z1] ^2+
    If [x1−1≤0, 0, x1−1] ^2+If [y1−1≤0, 0, y1−1] ^2+If [z1−1≤0, 0, z1−1] ^2);
w [x0__, y0__, z0__] :=Find Minimum [v [x1, y1, z1], {x1, x0, x0+0.1},
  {y1, y0, y0+0.1}, {z1, z0, z0+0.1}];
sol=Flatten [Table [w [x0, y0, z0], {x0, 0, 1.0, 0.5}, {y0, 0, 1.0, 0.5}, {z0, 0, 1.0, 0.5}], 2];
tol=1.0*10^(−10);
data=Table [If [sol [[i, 1]]<tol, i, "nonopt"], {i, 1, Length [sol]}];
ss=Cases [data, _Integer];
Table [sol [[ss [[i]]]], {i, 1, Length [ss]}]
```