産業動学に関する研究ノート（数値計算編）(2)

加　藤　　浩

5．コ　ー　ド

前節では，効率性の推移確率を特定化し，マルコフ完全ナッシュ均衡を導出する数値計算用の産業動学モデルを提示した。このモデルをもとにプログラム・コードを構築し，均衡値を算出する[29]。本節では，メインルーチンにおいて核となる部分のコードと，プログラムを構成するいくつかの関数（サブルーチン）のコードを記載し，解説を加える。なお，コードの行頭にある数字は行番号を表す。表１はコードで使用するパラメータを列挙したものである。これらのパラメータには，事前に具体的な数値を設定しておく[30]。表２にコードで使用する変数を列挙する。これらの変数はループを通じて値が変化する。マルコフ完全ナッシュ均衡を計算するアルゴリズムは，表３のようになっている（図１も参照せよ）。

29) 本稿で提示するコードは，
https://scholar.harvard.edu/pakes/pages/pakes-maguire-algorithm-0 を一部修正したものである。

30) 各パラメータを構造体変数のメンバに対応させて，具体的な数値を構造体メンバに代入した上で，別ファイルに保存しておく。例えば，`p.MAX_FIRMS = 3`，`p.KMAX = 19` のように数値を代入する。メインルーチンでこのファイルを呼び出して，構造体の各メンバをパラメータに代入する。例えば，`kmax = p.KMAX`，`rlnfirms = p.MAX_FIRMS` のように代入する。パラメータの数値は別ファイルから変更することができるので，メインルーチンを書き換えずに済む。

kmax：最大の効率性 $\overline{\omega}$
x_entryl：参入費用の下限 x_L^e
x_entryh：参入費用の上限 x_H^e
phi：スクラップ価値 ϕ
entry_k：参入企業の効率性 ω^e
rlnfirms：最大企業数 N
beta：割引因子 β
delta：ν が上昇する確率 δ
a：投資の効果 a
binom：二項係数行列 Binom（(rlnfirms＋kmax＋1)×(rlnfirm＋kmax＋2) 行列）

表１　パラメータの一覧

profit：利潤行列 Π（wmax×nfirms 行列）
dtable：クイック・デコーディング用のルックアップ・テーブル（nfirms×wmax 行列）
etable：クイック・エンコーディング用のルックアップ・テーブル（(kmax＋1)$^{\text{nfirms}}$ 次元ベクトル）
isentry：参入確率 λ（kmax 次元ベクトル）
nfirms：反復させる企業数 N_0
newvalue：価値関数行列（新しい値）$V^{(i)}$（wmax×nfirms 行列）
newx：投資関数行列（新しい値）$X^{(i)}$（wmax×nfirms 行列）
oldvalue：価値関数行列（古い値）$V^{(i-1)}$（wmax×nfirms 行列）
oldx：投資関数行列（古い値）$X^{(i-1)}$（wmax×nfirms 行列）
wmax：産業構造の総数 $\#\Omega$
prising：効率性が上昇する確率 $\Pr(\tau \mid x)$

表２　変数の一覧

(1) 二項係数行列 binom の構築
(2) nfirms ← 1
(3) 1 社問題の利潤データ profit を読み込む
(4) dtable の構築
(5) etable の構築
(6) 関数 update を呼び出す
(7) 反復
　norm＞tol かつ avgnorm＞0.001×tol である限り以下の処理を繰り返す
　(7-1) 関数 contract を呼び出す
　　(7-1-1) 関数 chkentry を呼び出す
　　(7-1-2) $w = 1, \cdots,$ wmax について関数 optimize を呼び出す
　(7-2) oldx ← newx
　(7-3) oldvalue ← newvalue
(8) 1 社問題の均衡データ newvalue, newx, prising, isentry を保存する
(9) nfirms ← nfirms + 1
(10) nfirms 社問題の利潤データ profit を読み込む
(11) mask の構築
(12) dtable の構築
(13) etable の構築
(14) 関数 update を呼び出す
(15) 反復
　norm＞tol かつ avgnorm＞0.0001×tol である限り以下の処理を繰り返す
　(15-1) 関数 contract を呼び出す
　　(15-1-1) 関数 chkentry を呼び出す
　　(15-1-2) $w = 1, \cdots,$ wmax について関数 optimize を呼び出す
　(15-2) oldx ← newx
　(15-3) oldvalue ← newvalue
(16) nfirms 社問題の均衡データ newvalue, newx, prising, isentry を保存する
(17) nfirms ← nfirms + 1
以下 nfirms = rlnfirms となるまで(10)～(17)を繰り返す

表3　アルゴリズム

5.1　メインループ

```
01    tol = 0.0001;
02
03    % binomの構築%
04
05    nfirms = 1;
06    while nfirms <= rlnfirms
07      wmax = binom(nfirms+1+kmax, kmax+2);
```

```
08
09      % nfirms社問題のprofitを呼び出す%
10      % maskの構築（nfirms>1のとき）%
11      % dtableの構築%
12      % etableの構築%
13
14      update;
15
16      norm = tol + 1;
17      avgnorm = norm;
18      while (norm > tol) && (avgnorm > 0.001*tol)
19
20        contract;
21
22        norm = max(max(abs(oldvalue - newvalue)));
23        avgnorm = mean(mean(abs(oldvalue - newvalue)));
24
25        oldx = newx;
26        oldvalue = newvalue;
27      end
28
29      w = kmax;
30      if nfirms > 1
31        w = [w; zeros(nfirms-1, 1)];
32      end
33      if max(newx(qencode(w):wmax, 1)) > 0
34      % 最大の効率性で投資水準が正になる-->再設定-->
                                          最大効率性の水準を増加させる%
35      end
```

```
36
37      prising = a.*newx./(1 + a.*newx);
38
39      save(['a.' PREFIX '_markov' int2str(nfirms) '.mat'], ...
40      'newvalue', 'newx', 'prising', 'isentry')
41
42      nfirms = nfirms + 1;
43    end
```

マルコフ完全ナッシュ均衡を求めるメイン・プログラムである[31]。プログラムを実行すると，rlnfirms 社問題の均衡価値関数 newvalue と均衡投資関数 newx が計算される[32]。このプログラムは次のような構造になっている。すなわち，産業で収容できる最大企業数 nfirms に関するループ（05～43行目）[33]の中に，均衡投資水準と均衡価値の推測値を改良するループ（16～27行目），および最大水準の効率性 kmax を持つ企業の投資水準が0となっているかどうかの条件判定（29～35行目）が含まれている。

ループの内部では次のような処理がなされる。いまループを一巡して，nfirms –1 社問題の均衡価値関数 oldvalue と均衡投資関数 oldx が導かれたとする。続いて nfirms 社問題のループに移る。まず，関数 update を呼び出し，oldvalue と oldx を nfirms 社問題の均衡に対する初期推測とする。次に関数 contract を呼び出し，古い推測値 oldvalue，oldx に対して反復処理を施し，新しい推測値 newvalue，newx を導く[34]。価値関数に関する停止条件が満たされるまで，関数 contract を何度も呼び出して推測値を更新する。停止条件が満たされ，反

31) C 言語の main 関数に相当する。

32) newvalue(w, n)は，エンコードされた産業構造 w と企業のインデックス n の組に対して，価値を対応させる関数である。同様に，newx(w, n)は投資水準を対応させる関数である。

33) MATLAB®では **while** でループさせずに **for** を用いて，

```
for nfirms = 1:rlnfirms
```

としたほうが効率は良いが，本稿では，ループ構造を明示するために **while** を用いたコードで記載する。

34) つまり，$V^{(i-1)}$ = oldvalue，$x^{(i-1)}$ = oldx，$V^{(i)}$ = newvalue，$x^{(i)}$ = newx となる。

復が終了したときに得られた newvalue と newx が，nfirms 社問題のマルコフ完全ナッシュ均衡の価値関数と投資関数となる。続いて nfirms + 1社問題のループに移る。以上が大まかな流れである。次に，各行のコードを解説していく。

01行目：許容誤差 tol を設定する。

03行目：二項係数行列 Binom を構築する（5.2節参照）。

07行目：(54)式からΩの濃度 wmax を計算する。wmax は企業数によって値が変わるので，nfirms が更新されるたびに計算し直す。

09行目：利潤行列 profit のデータが保存されているファイルを読み込む[35]。

10行目：mask を構築する（5.3節参照）。

11行目：ルックアップ・テーブル dtable を構築する（5.4節参照）。

12行目：ルックアップ・テーブル etable を構築する（5.5節参照）。

14行目：関数 update を呼び出して，nfirms – 1社問題の均衡値を nfirms 社問題の均衡値に対する初期推測とする（5.7節参照）。

18行目：停止条件が満たされるまで反復処理を繰り返す。

20行目：関数 contract を呼び出して反復処理を 1 回実行し，計算結果を newvalue，newx に格納する（5.6節参照）。

22行目：sup ノルムで測った誤差 norm を計算する。すなわち，

$$\texttt{norm} = \underset{n}{\mathrm{Max}}\,\underset{w}{\mathrm{Max}} \left| \texttt{oldvalue}(w,n) - \texttt{newvalue}(w,n) \right| . \qquad (101)$$

つまり，行列 oldvalue–newvalue の要素の中で，最大の値を誤差とする。

23行目：平均ノルムで測った誤差 avgnorm を計算する。すなわち，

$$\texttt{avgnorm} = \underset{n}{\mathrm{mean}}\,\underset{w}{\mathrm{mean}} \left| \texttt{oldvalue}(w,n) - \texttt{newvalue}(w,n) \right| . \qquad (102)$$

行列 oldvalue–newvalue の各行について平均をとると，nfirms 個の要素を持つ配列を得る。さらにこの配列の平均をとる。この値を誤差とする。

25，26行目：関数 contract で計算された newx, newvalue を，それぞれ oldx, oldvalue に上書きする。上書きされた oldx, oldvalue は，次の反復で関数 optimize を呼び出すときに参照される。

29～32行目：産業構造 w = (kmax, 0, …, 0)を考える。

---

35) 関数 profit を呼び出すと利潤行列が作成され，ファイルに保存される。関数 profit については 6 節参照。

33～35行目：w = (kmax, 0, …, 0)をエンコードする[36]。qencode(w)がエンコードされた産業構造である。qencode(w)～wmax の産業構造はすべて，1 番目の企業の効率性が kmax となっている。効率性は kmax より大きな値を取らないので，これらの産業構造における 1 番目の企業の投資水準は 0 になっていることが要求される。もしこの投資水準が正となるならば，kmax の数値を増加させたうえで再計算する必要がある。

37行目：(67)式をもとに，効率性が上昇する確率 prising を計算する。

39，40行目：均衡値のデータ newvalue，newx，prising，isentry をファイルに保存する[37]。

42行目：nfirms 社問題のループが完了したので，続いて nfirms + 1社問題を考える。rlnfirms 社問題の均衡値が計算されるまで，このプロセスを繰り返す。

5.2 Binom の構築

```
01    binom = eye(rlnfirms+kmax+1);
02    binom = [zeros(rlnfirms+kmax+1, 1), binom];
03
04    i = 2;
05    while i <= rlnfirms+kmax+1
06      binom(i, 2:i) = binom(i-1, 2:i) + binom(i-1, 1:i-1);
07      i = i+1;
08    end
```

二項係数行列 Binom を生成する。この行列は，エンコード関数 encode およびデコード関数 decode で使用される。エンコーディングに必要となる Binom の行

36) 計算の効率化のために，エンコーディングは関数 encode を直接呼び出すのではなく，クイック・エンコード関数 qencode を介して，事前に構築されたルックアップ・テーブルを参照する。

37) PREFIX はモデルのタイプを表すパラメータである。例えば，クールノー競争を PREFIX ='c'，ベルトラン競争を PREFIX ='b' と設定する。ベルトラン競争が展開される 3 社問題のデータは，ファイル名 a.b_markov3.mat に保存される。

数は kmax + rlnfirms + 1である[38]。したがって，Binom を kmax + rlnfirms + 1次単位行列で初期化する（01行目）。この単位行列に，kmax + rlnfirms + 1次元のゼロベクトルを１列目に付け加える（02行目）。(53)式に従って二項係数を計算し，binom の第２列から第 rlnfirms + kmax + 1列に値を格納する（04～08行目）。第１列は使用しないので，その要素は０のままにしておく（(52)式参照）。

5.3　mask の構築

```
01    two_n = 2^(nfirms-1);
02    mask = zeros(nfirms-1, two_n);
03
04    i = 0;
05    while i < two_n
06      msk = [];
07      j = 2;
08      k = i;
09      while j <= two_n
10        if mod(k, j) == 0
11          msk = [0; msk];
12        else
13          k = k - j/2;
14          msk = [1; msk];
15        end
16        j = j*2;
17      end
18      mask(:, i+1) = msk(1:nfirms-1);
19      i = i+1;
20    end
```

38) (58)式から，エンコーディングに用いる Binom の行番号は第ω_{nfirms} + 1 行～第ω_1 + nfirms 行である。nfirms は 1～rlnfirms の値を取り，ωは 0～kmax の値を取る。

maskは関数calcvalで使用される（5.10節参照）。効率性上昇のパラメータτは確率変数であり，その確率は投資水準によって決まる。自社を除いた各企業の実現値$(\tau_1, \cdots, \tau_{n-1}, \tau_{n+1}, \cdots, \tau_{\mathrm{nfirms}})$（$\tau_j = 0, 1$）について，すべてのパターンを列ごとに並べたものがmaskである。つまり，０か１のいずれかの値を取るnfirms－１個の数字の組を並べたものである[39]。組み合わせは$2^{\mathrm{nfirms}-1}$通りあるので，maskは$(\mathrm{nfirms}-1)\times 2^{\mathrm{nfirms}-1}$行列であり，次のような形をしている。

$$\mathtt{mask}=\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & 0 & 0 & & \vdots \\ \vdots & 0 & 1 & 1 & & \vdots \\ 0 & 1 & 0 & 1 & \cdots & 1 \end{pmatrix}. \tag{103}$$

行番号は企業のインデックスに対応する。ある列について，第j行の要素が０ならば，j番目の企業の効率性は上昇しない（つまり$\tau_j = 0$）ことを意味する。第j行の要素が１ならば，j番目の企業の効率性は上昇した（つまり$\tau_j = 1$）ことを意味する[40]。

nfirms－１次元ベクトルのmskを生成して，maskの各列を構築する。10進数i = 0, 1, …, $2^{\mathrm{nfirms}-1}-1$を２進数で表し，２進数の各桁の数字を桁の大きい順に第１要素から第nfirms－１要素へと代入したものがmskである。$\mathtt{i} = a_{\mathrm{nfirms}-1}2^{\mathrm{nfirms}-2}+\cdots+a_3 2^2 + a_2 2 + a_1$（$a_n = 0, 1$）と表すことができるとき，iの２進法表記は$a_{\mathrm{nfirms}-1}\cdots a_3 a_2 a_1$であり，

$$\mathtt{msk} = [a_{\mathrm{nfirms}-1}, \cdots, a_3, a_2, a_1]' \tag{104}$$

となる。$a_1, a_2, \cdots, a_{\mathrm{nfirms}-1}$の値は，次のループを実行することでiから取り出すことができる。i = 0, 1, …, $2^{\mathrm{nfirms}-1}-1$のループ内で（外側ループ）(04～20行目)，

39) 例えば，nfirms = 3 のとき，自社を除く 2 社のパターンをすべて列挙すると，
00, 01, 10, 11
となる。

40) 例えば，mask の第 1 列はすべての企業の効率性が変わらない，第 2 列は nfirms 番目の企業の効率性のみ上昇する，第 $2^{\mathrm{nfirms}-1}$ 列はすべての企業の効率性が上昇する，といった具合に解釈できる。

j = 2, 2^2, …, $2^{\mathrm{nfirms}-1}$のループを考え（内側ループ）(07～17行目)，i をj で割った余りを求める。ただし，k ← i と代入して，i の値は保持して，代わりにk の値を更新する（08行目)。j = 2からスタートして，k をj で割った余りを逐次求めていく。mod(k, j)はk をj で割ったときの余りであり，その値は0か1である。k を2で割ったとき，余りが0となるならば$a_1 = 0$であり，msk に0を格納する(11行目)。余りが1となるときは$a_1 = 1$であり，msk に1を格納し(14行目)，さらにk からj/2 = 1を引いて，

$$\mathrm{k} = a_{\mathrm{nfirms}-1}2^{\mathrm{nfirms}-2} + \cdots + a_3 2^2 + a_2 2 \tag{105}$$

とk を更新する（13行目）[41]。

次にj = 2^2とする（16行目)。更新されたk を2^2で割ったときの余りが0となるならば$a_2 = 0$なので，msk に0を付け加える（11行目)。余りが1のときは$a_2 = 1$であり，msk に1を付け加え（14行目)，さらにk からj/2 = 2を引いて，

$$\mathrm{k} = a_{\mathrm{nfirms}-1}2^{\mathrm{nfirms}-2} + \cdots + a_3 2^2 \tag{106}$$

とk を更新する（13行目)。続いてj = 2^3として（16行目)，同様の計算を繰り返す。

一般的には次のようになる。いまj = 2^hまで処理が済んでおり，かつ，

$$\mathrm{k} = a_{\mathrm{nfirms}-1}2^{\mathrm{nfirms}-2} + \cdots + a_{h+2}2^{h+1} + a_{h+1}2^h \tag{107}$$

と更新されているとする。k をj = 2^{h+1}で割った余りが0のとき，$a_{h+1} = 0$であるので，

$$\mathrm{k} = a_{\mathrm{nfirms}-1}2^{\mathrm{nfirms}-2} + \cdots + a_{h+2}2^{h+1}. \tag{108}$$

他方，k をj = 2^{h+1}で割った余りが1のとき$a_{h+1} = 1$であり，k を次のように更新する。

41) 余りが0でも1でも，k の値は(105)式になる。

$$\begin{aligned} \mathtt{k} &\leftarrow \mathtt{k} - \mathtt{j}/2 \\ &= (a_{\mathtt{nfirms}-1}2^{\mathtt{nfirms}-2} + \cdots + a_{h+2}2^{h+1} + 2^{h}) - 2^{h+1}/2 \\ &= a_{\mathtt{nfirms}-1}2^{\mathtt{nfirms}-2} + \cdots + a_{h+2}2^{h+1}. \end{aligned} \tag{109}$$

余りがいずれの場合でも k は同じ値となり，この k を $\mathtt{j} = 2^{h+1}$で割った余りが a_{h+2}となる。

$\mathtt{j} = 2^{\mathtt{nfirms}-1}$まで処理が完了すると i の内側ループは終了し，生成された msk を mask の第 i + 1列に格納する（18行目）。i の値を 1 つ増やして（19行目），j = 2に戻り，再び内側ループの処理を行う。$\mathtt{i} = 2^{\mathtt{nfirms}-1} - 1$まで到達し，外側ループが終了すると，mask が完成する。

5.4　dtable の構築

```
01    dtable = zeros(nfirms, wmax);
02
03    i = 1;
04    while i <= wmax
05      dtable(:, i) = decode(i);
06      i = i+1;
07    end
```

クイック・デコーディングで使用するルックアップ・テーブル dtable を作成する。nfirms の値が更新されるたびに wmax の値が変わるため，dtable も作り直す必要がある（01行目）。エンコードされた産業構造 i = 1, …, wmax を関数 decode に入力して，デコードされた産業構造ベクトルを dtable の第 i 列に格納する（04～07行目）。

5.5　etable の構築

```
01    oneton = [1:nfirms]';
02
```

```
03   if nfirms <= encfirm;
04     multfac = (kmax+1).^(oneton-1);
05     etable = zeros((kmax+1)^nfirms, 1);
06
07     i = 0;
08     while i < size(etable, 1)
09       msk = [];
10       k = i;
11
12       j = kmax+1;
13       while j <= size(etable, 1)
14         msk = [mod(k, j)*(kmax+1)/j; msk];
15         k = k - (mod(k, j));
16         j = j*(kmax+1);
17       end
18
19       etable(i+1) = encode(flipud(sortrows(msk(1:nfirms), 1)));
20       i = i+1;
21     end
22   end
```

クイック・エンコーディングで使用するルックアップ・テーブル etable を作成する。ただし，エンコードする企業の数が多いときは etable を使用しない[42]。ルックアップ・テーブルにエンコードできる最大の企業数を encfirm とする。nfirms ≤ encfirm のときに，etable を用いてエンコーディングを実行する。

nfirms 次元ベクトル msk を生成して，etable の各列を構築する。産業構造ベクトル ntuple = $(\omega_1, \cdots, \omega_{\text{nfirms}})$の取りうる値の組み合わせは$(\text{kmax}+1)^{\text{nfirms}}$

42) Pakes, Gowrisankaran and McGuire のコードでは，etable を 2 つに分けてエンコーディングを行っている。

通りある。これらの組み合わせに対して，0から$(\mathtt{kmax}+1)^{\mathtt{nfirms}}-1$までの番号を重複しないように割り振る。$\mathtt{kmax}+1$の累乗を並べたベクトルを multfac とする（04行目）。

$$\mathtt{multfac}=\begin{pmatrix}1\\ \mathtt{kmax}+1\\ (\mathtt{kmax}+1)^2\\ \vdots\\ (\mathtt{kmax}+1)^{\mathtt{encfirm}-1}\end{pmatrix}. \tag{110}$$

ntuple と multfac の内積が，ntuple に割り振る番号 i = 0, 1, …, $(\mathtt{kmax}+1)^{\mathtt{nfirms}}-1$である。つまり，

$$\mathtt{i}=\mathtt{ntuple}\cdot\mathtt{multfac}'. \tag{111}$$

割り振られた番号 i を kmax + 1進数に変換して，変換した数の各桁の数字を桁の大きい順に msk の第1要素から第 nfirms 要素へと格納していく。$\mathtt{i}=\omega_{\mathtt{nfirms}}(\mathtt{kmax}+1)^{\mathtt{nfirms}-1}+\cdots+\omega_3(\mathtt{kmax}+1)^2+\omega_2(\mathtt{kmax}+1)+\omega_1$（$\omega_h=0, 1, \cdots, \mathtt{kmax}$）のように表すことができるとき，i の $\mathtt{kmax}+1$進法表記は$\omega_{\mathtt{nfirms}}\cdots\omega_3\omega_2\omega_1$となる。したがって，

$$\mathtt{msk}=[\omega_{\mathtt{nfirms}}, \cdots, \omega_3, \omega_2, \omega_1]' \tag{112}$$

となる。$\omega_1, \cdots, \omega_{\mathtt{nfirms}}$の値は，次のループを実行することで i から取り出すことができる。$\mathtt{i}=0, 1, \cdots, (\mathtt{kmax}+1)^{\mathtt{nfirms}}-1$のループ内で（外側ループ）（07～21行目），$\mathtt{j}=\mathtt{kmax}+1, (\mathtt{kmax}+1)^2, \cdots, (\mathtt{kmax}+1)^{\mathtt{nfirms}}$のループを考え（内側ループ）（12～17行目），i を j で割った余りを順次求める。ただし，$\mathtt{k}\leftarrow\mathtt{i}$として（10行目），i の値を保持しておく。k を $\mathtt{j}=\mathtt{kmax}+1$で割った余りは，

$$\mathrm{mod}(\mathtt{k}, \mathtt{kmax}+1)=\omega_1 \tag{113}$$

となるから，

$$\begin{aligned}\mathrm{mod}(\mathtt{k}, \mathtt{j})\times(\mathtt{kmax}+1)/\mathtt{j}&=\mathrm{mod}(\mathtt{k}, \mathtt{kmax}+1)\times(\mathtt{kmax}+1)/(\mathtt{kmax}+1)\\ &=\omega_1\end{aligned} \tag{114}$$

と計算して，計算結果のω_1を msk に格納する（14行目）。さらに，k を以下のように更新する（15行目）[43]。

$$\begin{aligned} \mathtt{k} &\leftarrow \mathtt{k} - \mathrm{mod}(\mathtt{k}, \mathtt{j}) \\ &= \mathtt{k} - \omega_1 \\ &= \omega_{\mathtt{nfirms}}(\mathtt{kmax}+1)^{\mathtt{nfirms}-1} + \cdots + \omega_3(\mathtt{kmax}+1)^2 + \omega_2(\mathtt{kmax}+1). \end{aligned} \quad (115)$$

また，j = (kmax + 1)^{2}と更新する（16行目）。続いて，更新された k を更新された j で割り，余りを求める。

$$\mathrm{mod}(\mathtt{k}, (\mathtt{kmax}+1)^2) = \omega_2(\mathtt{kmax}+1) \quad (116)$$

となるから，

$$\begin{aligned} \mathrm{mod}(\mathtt{k}, \mathtt{j}) \times (\mathtt{kmax}+1) / \mathtt{j} &= \mathrm{mod}(\mathtt{k}, (\mathtt{kmax}+1)^2) \times (\mathtt{kmax}+1) / (\mathtt{kmax}+1)^2 \\ &= \omega_2 \end{aligned} \quad (117)$$

と計算して，計算結果のω_2を msk に追加する（14行目）。さらに，k を以下のように更新する（15行目）。

$$\begin{aligned} \mathtt{k} &\leftarrow \mathtt{k} - \mathrm{mod}(\mathtt{k}, \mathtt{j}) \\ &= \omega_{\mathtt{nfirms}}(\mathtt{kmax}+1)^{\mathtt{nfirms}-1} + \cdots + \omega_3(\mathtt{kmax}+1)^2. \end{aligned} \quad (118)$$

また，j = (kmax + 1)^{3}と更新する（16行目）。再び k を j で割った余りを求めて，ω_3を取り出す。

一般的には次のようになる。いま，j = (kmax + 1)^h，および，

$$\mathtt{k} = \omega_{\mathtt{nfirms}}(\mathtt{kmax}+1)^{\mathtt{nfirms}-1} + \cdots + \omega_{h+1}(\mathtt{kmax}+1)^h + \omega_h(\mathtt{kmax}+1)^{h-1} \quad (119)$$

と更新されているとする。k を j で割った余りは，

$$\mathrm{mod}(\mathtt{k}, (\mathtt{kmax}+1)^h) = \omega_h(\mathtt{kmax}+1)^{h-1} \quad (120)$$

---

43) k = 0 − mod(0, j) = 0 なので，k を更新したときに k = 0 となると，以降の msk の要素はすべて 0 になる。

であるから，

$$\begin{aligned}\mathrm{mod}(\mathtt{k},\mathtt{j})\times(\mathtt{kmax}+1)/\mathtt{j} &= \mathrm{mod}(\mathtt{k},(\mathtt{kmax}+1)^{h})\times(\mathtt{kmax}+1)/(\mathtt{kmax}+1)^{h}\\ &= \omega_h \end{aligned} \tag{121}$$

と計算して，計算結果のω_hを msk に追加する。さらに，k を以下のように更新する。

$$\begin{aligned}&\mathtt{k} \leftarrow \mathtt{k} - \mathrm{mod}(\mathtt{k},\mathtt{j})\\ &\quad = \omega_{\mathtt{nfirms}}(\mathtt{kmax}+1)^{\mathtt{nfirms}-1} + \cdots + \omega_{h+1}(\mathtt{kmax}+1)^{h}.\end{aligned} \tag{122}$$

また，j = (kmax + 1)$^{h+1}$と更新する。このような処理を繰り返していき，j = (kmax + 1)$^{\mathtt{nfirms}}$，k = $\omega_{\mathtt{nfirms}}$(kmax + 1)$^{\mathtt{nfirms}-1}$まで到達したら内側ループは完了する。

こうして導かれた msk の要素を降順に並べ替え[44]，関数 encode に入力すると，ある自然数が出力される。この自然数は，番号 i が割り振られた産業構造ベクトルをエンコードしたものであり，etable の第 i + 1要素に格納する（19行目）[45]。

続いて i を 1 つ増やして（20行目），別の産業構造ベクトルをエンコードする。i = (kmax + 1)$^{\mathtt{nfirms}}$ − 1まで繰り返すと etable が完成する。

---

44) 関数 sortrows(A, n)は，第 n 列の要素を基準として，行列 A を昇順に並べ替える MATLAB®の組み込み関数である。flipud は行列の行を上下に反転する組み込み関数である。したがって，flipud(sortrows(A, n))は行列 A を降順に並べ替える。なお，MATLAB®の最新のバージョンでは，sort(A, n, 'descend')で降順に並べ替えることができる。

45) 例えば，[0, 1, 0, ⋯, 0]と[0, 0, 1, 0, ⋯, 0]に割り振られる番号は異なるが，ともに[1, 0, ⋯, 0]と同じ産業構造である。したがって，これらを関数 encode に入力すると，戻り値はすべて同じになる。ゆえに，etable は値が重複する要素を持つ。

番号	0	1	2	…	kmax	kmax＋1	kmax＋2	…	2kmax＋1	2kmax＋2
産業構造	$\begin{bmatrix}0\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\1\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\2\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\\mathrm{kmax}\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\1\\0\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\1\\1\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\1\\\mathrm{kmax}\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\2\\0\end{bmatrix}$

番号	2kmax＋3	…	3kmax＋2	3(kmax＋1)	…	kmax(kmax＋1)	…
産業構造	$\begin{bmatrix}0\\\vdots\\0\\2\\1\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\2\\\mathrm{kmax}\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\3\\0\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\\mathrm{kmax}\\0\end{bmatrix}$	…

番号	kmax(kmax＋1)＋kmax	$(\mathrm{kmax}+1)^2$	…	$\mathrm{kmax}(\mathrm{kmax}+1)^2$	…	$(\mathrm{kmax}+1)^2-1$
産業構造	$\begin{bmatrix}0\\\vdots\\0\\\mathrm{kmax}\\\mathrm{kmax}\end{bmatrix}$	$\begin{bmatrix}0\\\vdots\\0\\1\\0\\0\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\\mathrm{kmax}\\0\\0\end{bmatrix}$	…	$\begin{bmatrix}0\\\vdots\\0\\\mathrm{kmax}\\\mathrm{kmax}\\\mathrm{kmax}\end{bmatrix}$

番号	$(\mathrm{kmax}+1)^2$	$(\mathrm{kmax}+1)^{\mathrm{nfirms}-1}$	…	$\mathrm{kmax}(\mathrm{kmax}+1)^{\mathrm{nfirms}-1}$	…	$(\mathrm{kmax}+1)^{\mathrm{nfirms}}-1$
産業構造	$\begin{bmatrix}0\\\vdots\\1\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}$	…	$\begin{bmatrix}\mathrm{kmax}\\0\\\vdots\\0\end{bmatrix}$	…	$\begin{bmatrix}\mathrm{kmax}\\\vdots\\\mathrm{kmax}\end{bmatrix}$

表４　産業構造ベクトルに割り振られる番号

5.6 関数 contract

```
function [] = contract()
  global newvalue newx wmax

  chkentry;

  w = 1;
  while w <= wmax;
    [newx(w, :), newvalue(w, :)] = optimize(w);
    w = w+1;
  end
end
```

関数 contract を実行すると，均衡価値および均衡投資水準の改良された推測値を計算して，newvalue，newx に代入する。関数 contract は次の２つの処理を行う。まず，関数 chkentry を呼び出して，産業構造 w = 1, …, wmax における参入確率を計算する。次に，産業構造 w = 1, …, wmax に対して関数 optimize を呼び出し，クーン＝タッカー条件を満たす投資水準とベルマン方程式を満たす価値を，すべての企業について求めて，newx, newvalue の第 w 行に代入する。

5.7 関数 update

```
function [] = update()
  global isentry nfirms wmax newvalue newx oldvalue oldx
  oldx = zeros(wmax, nfirms);
  oldvalue = zeros(wmax, nfirms);

  if nfirms == 1
    i = 1;
    while i <= wmax
```

```
09          oldvalue(i, :) = 1 + 0.1*i;
10          i = i + 1;
11        end
12
13      else
14        w = 1;
15        while w <= wmax;
16          tuple = qdecode(w);
17          nfirms = nfirms-1;
18          n = encode(tuple(1:nfirms));
19          oldx(w, 1:nfirms) = newx(n, 1:nfirms);
20          oldvalue(w, 1:nfirms) = newvalue(n, 1:nfirms);
21          nfirms = nfirms + 1;
22          tuple(nfirms-1) = tuple(nfirms);
23          tuple(nfirms) = 0;
24          oldvalue(w, nfirms) = oldvalue(encode(tuple), nfirms-1);
25          oldx(w, nfirms) = oldx(encode(tuple), nfirms-1);
26          w = w+1;
27        end
28      end
29
30      isentry = zeros(wmax, 1);
31      newx = zeros(wmax, nfirms);
32      newvalue = zeros(wmax, nfirms);
33    end
```

nfirms － 1社問題で導かれた均衡投資関数 newx, および均衡価値関数 newvalue を，(47)，(48) 式の設定に従い，それぞれ oldx, oldvalue に代入する。この oldx, oldvalue を nfirms 社問題の均衡に対する初期推測とする。

したがって，newx, newvalue をグローバル変数として宣言し，この関数から参照する。

nfirms＝1のときは，

$$\texttt{oldvalue}=\begin{pmatrix}1+0.1\\1+0.2\\\vdots\\1+0.1\texttt{wmax}\end{pmatrix} \tag{123}$$

とする（06～11行目）。また，

$$\texttt{oldx}(w,1)=0\quad(w=1,\cdots,\texttt{wmax}) \tag{124}$$

とする（03行目）。

投資関数newxと価値関数newvalueは，nfirms－1次元の産業構造ベクトルをエンコードしたものを引数に取る。それゆえ，産業構造($\omega_1,\cdots,\omega_{\texttt{nfirms}}$)のうち，要素をnfirms－1個だけ取り出す必要がある。(47)，(48）式に従い，1番目～nfirms－1番目の企業については，($\omega_1,\cdots,\omega_{\texttt{nfirms}-1}$)とする。nfirms番目の企業については，($\omega_1,\cdots,\omega_{\texttt{nfirms}-2},\omega_{\texttt{nfirms}}$)とする。具体的には初期推測を次のように設定する。

16行目：wをデコードして，産業構造ベクトルtupleを得る[46]。tupleはnfirms次元ベクトルであり，tuple＝($\omega_1,\cdots,\omega_{\texttt{nfirms}-1},\omega_{\texttt{nfirms}}$)とする。

17行目：nfirmsをnfirms－1に置き換えて，企業数をnfirms－1で考える。

18行目：tupleの最後の要素を除いたベクトル($\omega_1,\cdots,\omega_{\texttt{nfirms}-1}$)をエンコードする。これをnとする。

19，20行目：1番目からnfirms－1番目の企業について，newx，newvalueに産業構造nを入力し，出力されたものをoldx，oldvalueに代入する。このように設定したoldx，oldvalueを，産業構造wにおける均衡値の初期推測とする。

---

46) デコーディングは，関数decodeに直接入力することは避けて，クイック・デコード関数qdecodeに入力して，事前に構築されたルックアップ・テーブルを参照するという方法を取る。

21行目：企業数を元の nfirms に戻す。

22行目：nfirms－1番目の企業の効率性をω_{nfirms}とする。

23行目：nfirms 番目の企業の効率性を0とする。したがって，tuple = (ω_1, …, $\omega_{\text{nfirms}-2}$, ω_{nfirms}, 0)となる。

24，25行目：nfirms番目の企業については，19，20行目で計算したnfirms－1番目の企業のoldx，oldvalueを初期推測として使い回すが，encode(tuple)を産業構造として入力する。

31, 32行目：nfirms社問題に使用するため，newx，newvalueを初期化する。

すべての産業構造w = 1, …，wmaxについて，上記のように初期推測を設定する。

5.8　関数 optimize

```
01   function [out1, out2] = optimize(w)
02     global a beta entry_k isentry nfirms oldvalue oldx phi profit
03     locw = qdecode(w);
04     locwx = locw;
05     oval = oldvalue(w, :)';
06     ox = oldx(w, :)';
07     nval = zeros(nfirms, 1);
08     nx = zeros(nfirms, 1);
09
10     [m, ind] = min(oval);
11     i = (m == phi)*(ind-1) + (m > phi)*nfirms;
12
13     if i < nfirms
14       locwx(i+1:nfirms) = zeros(nfirms-i, 1);
15     end
16
17     entered = isentry(qencode(flipud(sortrows(locwx, 1))));
18     locwe = locwx;
```

```
19      locwe(nfirms) = entry_k;
20
21      j = 1;
22      while j <= nfirms
23        if locw(j) == 0;
24          nval(j:nfirms) = phi*ones(nfirms-j+1, 1);
25          break
26        end
27
28        v1 = 0;
29        v2 = 0;
30
31        if entered < 1
32          [v1, v2] = calcval(j, locwx, ox, locw(j));
33        end
34
35        if entered > 0
36          [tempv1, tempv2] = calcval(j, locwe, ox, locw(j));
37          v1 = entered*tempv1 + (1-entered)*v1;
38          v2 = entered*tempv2 + (1-entered)*v2;
39        end
40
41        if v1 <= v2
42          r = 1.0;
43        else
44          r = 1.0/(beta*a*(v1 - v2));
45        end
46
47        r = min([max([r; 1e-13]); 1]);
```

```
48        p = 1.0 - sqrt(r);
49        nx(j) = p/(a - a*p);
50        nval(j) = profit(w, j) - nx(j) + beta*(v1*p + v2*(1-p));
51
52        if nval(j) <= phi
53          nval(j) = phi;
54          nx(j) = 0;
55        end
56
57        if (j < nfirms) && (nval(j) == phi)
58          nval(j+1:nfirms) = ones(nfirms-j, 1)*phi;
59          break;
60        end
61
62        ox(j) = nx(j);
63        locwx(j) = (nval(j) > phi)*locw(j);
64        locwe(j) = locwx(j);
65        j = j+1;
66      end
67
68      out1 = nx';
69      out2 = nval';
70    end
```

関数 optimize は，エンコードされた産業構造 w を引数に取り，w における均衡価値の新しい推測値 nval，および均衡投資水準の新しい推測値 nx を返す[47]。

47) 反復式(71)式に従って計算する。ox は $(x_1^{(i)},\cdots,x_{n-1}^{(i)},x,x_{n+1}^{(i-1)},\cdots x_N^{(i-1)})$，oval は $V^{(i-1)}(\cdot,n)$，nx(n)は $x_n^{(i)}$，nval(n)は $V^{(i)}(\cdot,n)$にそれぞれ対応する。効率性上昇の確率 p_up を通じて，他社の投資水準が自社の期待割引現在価値に影響を与える。

前回の反復で計算された推測値 oldval, oldx をグローバル変数として宣言して，この関数から参照できるようにする。

反復処理を実行するたびに，関数 optimize は繰り返し呼び出される。前回の反復で退出した企業は今回の反復でも退出しているものとし，空きスロットが存在するならば，ある確率で参入が発生する。したがって，入力された産業構造 w を更新して，退出と参入を考慮に入れた新たな産業構造の下での投資水準を求める。

03行目：w をデコードして locw を出力する。

05，06行目：w における各企業の oldvalue, oldx の値を並べたベクトルを，それぞれ oval，ox とおく。

10行目：oval の最小値を m，その値を持つ企業のインデックスを ind とする。

11行目：前回の反復で産業に残っている企業数 i を求める。すなわち，oval の要素のうち，スクラップ価値よりも大きい値を取る要素の総数である。ここで，自社よりも高い効率性を持つ企業が退出するときは，自社も退出するという退出政策を前提としている。また，インデックスが大きい企業ほど効率性が低くなるように産業構造の集合を制限している。このことから，oval の最小値がスクラップ価値 phi に等しいとき（m = phi）は，ind 番目～nfirms 番目の企業が退出している[48]。したがって，産業には ind − 1だけの企業が活動している。価値の最小値がスクラップ価値より大きいとき（m > phi）は，nfirms の企業すべてが産業で活動している。

13～15行目：退出が完了した後の産業構造 locwx を作る。産業で活動している企業数 i が nfirms よりも少ないときは，i + 1番目～nfirms 番目の企業が退出しているので，その効率性を0とする。i = nfirms のときは退出する企業がいないので，04行目の設定は変更されず，locwx = locw となる。

17行目：locwx の要素を降順に並べ替えて関数 isentry に入力すると，参入確率 entered が出力される。

48) 前回の反復で退出する企業の価値を phi としているので（53 行目），m < phi は起こり得ない。

18，19行目： locwx の第 nfirms 要素に参入企業の効率性 entry_k を代入して，locwe と置く。これは，参入発生後の産業構造である[49]。

j = 1, …, nfirms 番目の企業の投資水準と価値を求める。1 番目の企業から始めて，nfirms 番目まで順番に計算していく。他社の投資水準は ox であると予測する（(49) 式参照)。ただし，自社のインデックスを n とすると，1 番目～n – 1番目の企業の投資水準は，すでに関数 optimize で更新されている[50]。このような前提を踏まえて，期待割引現在価値を計算し，これを最大にする投資水準 nx を決める。最大化された期待割引現在価値が nval である。

23～26行目：w において j 番目の企業の効率性が0ならば，この企業は退出している。したがって，j 番目～nfirms 番目の企業の価値を phi とする。投資水準は08行目で初期化した0がそのまま維持される。すべての企業の価値と投資水準を計算したので，**while** ループから抜ける。

31～33行目：投資実行後に期待できる将来価値 Calcval を計算する[51]。そのために必要となる値を求める。参入が発生しない確率1 – entered が正のときは，参入がないときの将来価値 v1, v2を求める必要がある。j 番目の企業の効率性は locw(j)，産業構造は locwx，各企業の投資水準は ox である。これらを関数 calcval に入力すると，v1, v2が出力される。v1は自社の効率性が上昇したときの価値，v2は効率性が変わらないときの価値である[52]。1 – entered ≤ 0のときは，参入が確実に発生する。このときは，31～33行目は実行されないため，28, 29行目で設定した v1 = 0，v2 = 0となる。

---

49) locw = (ω_1, …, ω_{nfirms})とすると，locwx = (ω_1, …, ω_k, 0, …, 0), locwe = (ω_1, …, ω_k, 0, …, 0, entry_k)となる。関数 calcval は locwe を降順に並べるので，entry_k の位置が変わる。

50) nx は他社の投資水準 ox に対する最適反応となる。関数 optimize は最適反応法と呼ばれる。

51) 参入企業の効率性が entry_k である産業構造 locwe は，参入が決定された次の期に実現する（図２参照)。この期以降に既存企業が得る期待割引現在価値が，(77)式で定義される Calcval である。

52) (77)式の E$V(\boldsymbol{\omega}' + \boldsymbol{\tau} - \nu i, n)$に相当する。E は$\tau_1$, …, τ_{n-1}, τ_{n+1}, …, τ_{nfirms}, νについて期待値を取る。

35, 36行目：entered が正のときは，参入が発生したときの将来価値 tempv1, tempv2を求めておく。これらの価値は，j 番目の企業の効率性 locw(j)，産業構造 locwe, 各企業の投資水準 ox を関数 calcval に入力することで計算される。tempv1は自社の効率性が上昇するときの価値，tempv2は効率性が変わらないときの価値である[53]。

37, 38行目：(77) 式で定義される Calcval を計算する。確率 entered で参入が起こり，tempv1を得る。他方，確率1 – entered で参入は起こらず，v1を得る。これらの期待値を取り，v1に上書きする。v2についても同様に計算する[54]。

41～45行目：(88) 式で定義される r を計算する。ただし，v1 ≤ v2のときは，r が負となるか，あるいはゼロ除算をするため，r ＝1としておく。

47, 48行目：$10^{-13} \le r \le 1$として効率性上昇の確率を計算する。v1が v2と比べて極めて大きいため，実際の r の値が計算機イプシロンよりも小さくなるならば，$r = 10^{-13}$としておく[55]。

49行目：期待割引現在価値を最大にする投資水準，すなわち (90) 式を満たす投資水準 nx を計算する。

50行目：(91) 式から，最大化された期待割引現在価値 nval を計算する。自社の効率性が上昇するならば v1，効率性が同じ水準のままならば v2の将来価値を，投資実行後に得る。期待値を取り，割り引いてから純収益 profit – nx に加えると nval が求まる。

52～55行目：nval が phi 以下ならば，投資をせずに退出する。このときは，nval を phi で上書きし，投資水準を 0 とする。

57～60行目：j 番目の企業が退出するならば，j + 1番目～nfirms 番目の企業も退出するので，これらの企業の価値を phi とする。08行目の初期化が保持され，これらの企業の投資水準は 0 となる[56]。すべての企業について投資水準と価値を

---

53) (77)式の $EV(\boldsymbol{\omega}^{\dagger} + \omega^{e}\boldsymbol{e}_{n_e} + \boldsymbol{\tau} - \nu\boldsymbol{i}, n)$ に相当する。

54) v1 は(83)式，v2 は(84)式を計算したものである。

55) 計算機イプシロンをε_Mとすると，実際の r が $r < \varepsilon_M$ となるならば，計算機では 1 + r = 1 となり，(89)式は p = 1 と計算される。この演算により，少なくともε_Mの丸め誤差が生じる。MATLAB®では$\varepsilon_M = 2.2204 \times 10^{-16}$ となる。

56) これらの企業の中には前回の反復では退出しなかったが，今回の反復では他社の投資水準が更新され，産業構造も変化したので，退出を選択する企業が含まれる。

計算したので，**while** ループから抜ける。

62行目：以上で導かれた企業 j の投資水準 nx(j)を ox(j)に上書きする。したがって，次の j + 1番目の企業に関するループでは，j 番目の企業の投資水準は関数 optimize で更新されたものが使用される。

63行目：j 番目の企業の新しい効率性を代入して，退出を考慮に入れた産業構造 locwx を更新する。したがって，j + 1番目の企業に関するループでは，この更新された産業構造の下で投資水準 nx(j + 1)を決める。nval(j) > phi ならば，j 番目の企業は産業で活動している。ゆえに，locwx における j 番目の企業の効率性は locw(j)となる。そうでない場合は，退出しているため，効率性を 0 とする。

64行目：locwe を更新する。j 番目の企業が退出するならば，その効率性を 0 に変える。

65行目：j + 1番目の企業に関するループに移る。

68，69行目：すべての企業について計算した nx, nval を出力する[57]。

5.9　関数 chkentry

```
01    function [] = chkentry()
02      global beta entry_k isentry nfirms wmax oldx x_entryl x_entryh
03
04      w = 1;
05      while w <= wmax
06        locw = qdecode(w);
07        if locw(nfirms) == 0
08          [~, v1] = calcval(nfirms, locw, oldx(w, :)', entry_k);
09          val = beta*v1;
10          isentry(w) = (val - x_entryl)/(x_entryh - x_entryl);
```

57) 関数 contract ですべての w = 1, …, wmax について nx, nval を計算して，newx, newvalue に格納する。メインループでは，newx, newvalue をそれぞれ oldx, oldvalue に代入する。関数 optimize は，この oldx, oldvalue をグローバル変数として呼び出して値を改良する。

```
11        end
12        w = w+1;
13      end
14
15      isentry = min([isentry, ones(wmax, 1)]')';
16      isentry = max([isentry, zeros(wmax, 1)]')';
17    end
```

産業構造 w = 1, …, wmax における参入確率 isentry を計算する。まず，w をデコードして locw を得る（06行目）。nfirms 番目のスロットに空きがあれば，ある確率で参入が発生する（07行目）[58]。産業構造が locw，各企業の投資水準が oldx であるとき，効率性 entry_k を持つ参入企業の将来価値を計算する。これらの変数を引数として関数 calcval に入力する。参入企業の効率性は上昇しないので，calcval の２番目の戻り値のみを考える（08行目）。参入決定から１期間経った後にこの価値を獲得するので，割り引いたものが参入価値 val となる（09行目）。(100) 式に従って参入確率 isentry を計算し（10行目），0 ≤ isentry ≤ 1となるように調整する（15，16行目）。

5.10　関数 calcval

```
01    function [out1, out2] = calcval(place, w, x, k)
02      global a delta kmax mask nfirms oldvalue two_n
03      z1 = zeros(nfirms, 1);
04      z2 = kmax*ones(nfirms, 1);
05
```

58) 参入確率の初期化は，isentry = [] と空の配列を設定しても差し支えない。w = 1 から出発して参入確率を計算する。この産業構造は，すべてのインデックスについて空きスロットとなっているので，07 行目の **if** 文は必ず真となる。また，1 社問題について，w = 1 以外の産業構造では独占企業が産業で活動しているので，参入確率はゼロとなる。このことは，2 社問題で w = 2, …, wmax の参入確率が 0 に初期化されていることに等しい。したがって，07 行目の **if** 文が実行されないならば，参入確率はゼロとなる。

```
06      if nfirms > 1
07        if place == 1
08          locmask = [zeros(1, two_n); mask];
09        elseif place == nfirms
10          locmask = [mask; zeros(1, two_n)];
11        else
12          locmask = [mask(1:place-1, :); zeros(1, two_n); ...
13            mask(place:nfirms-1, :)];
14        end
15      else
16        locmask = zeros(1, 1);
17      end
18
19      x(place) = 0;
20      w(place) = k;
21      justone = zeros(nfirms, 1);
22      justone(place) = 1;
23      p_up = (a.*x)./(1 + a.*x);
24
25      valA = 0;
26      valB = 0;
27
28      i = 1;
29      while i <= two_n
30        probmask = prod(2.*locmask(:, i).*p_up + 1 - locmask(:, i) - p_up);
31
32        d = w+locmask(:, i);
33        temp = flipud(sortrows([d, justone], 1));
34        d = temp(:, 1);
```

```
35      e = d-1;
36      e = max(([e, z1])')';
37      d = min(([d, z2])')';
38      [~, pl1] = max(temp(:, 2));
39
40      valB = valB + ((1-delta)*oldvalue(qencode(d), pl1) ...
41        + delta*oldvalue(qencode(e), pl1))*probmask;
42
43      d = w + locmask(:, i) + justone;
44      temp = flipud(sortrows([d, justone], 1));
45      d = temp(:, 1);
46      e = d-1;
47      e = max(([e, z1])')';
48      d = min(([d, z2])')';
49      [~, pl1] = max(temp(:, 2));
50
51      valA = valA + ((1-delta)*oldvalue(qencode(d), pl1) ...
52        + delta*oldvalue(qencode(e), pl1))*probmask;
53
54      i = i+1;
55    end
56
57    out1 = valA;
58    out2 = valB;
59  end
```

各企業が投資を実行した後に，τおよびνの値が実現し，効率性の水準が投資前と比べて増減する。それに合わせて産業構造も現在の$\boldsymbol{\omega}$から変化する。投資によって起こり得る産業構造の変化について，それらが実現する確率を考慮に入れて期

待価値を計算する。つまり，投資を実行することで期待される将来価値

$$\texttt{valA or valB} = \sum_{\tau_1=0}^{1}\cdots\sum_{\tau_{n-1}=0}^{1}\sum_{\tau_{n+1}=0}^{1}\cdots\sum_{\tau_{\texttt{nfirms}}=0}^{1}\left\{\sum_{\nu=0}^{1}\Pr(\nu)V(\boldsymbol{\omega}+\boldsymbol{\tau}-\nu\boldsymbol{i},n)\right\}\times\Pr(\tau_1)\times\cdots\times\Pr(\tau_{n-1})\times\Pr(\tau_{n+1})\times\cdots\times\Pr(\tau_{\texttt{nfirms}}) \tag{125}$$

を計算する関数が calcval である[59]。ただし，n は自社のインデックスであり，τ_n の値を所与としている。関数 calcval の引数は，自社のインデックス place，産業構造ベクトル w，各企業の投資水準のベクトル x，自社の効率性 k である。戻り値は将来価値 valA, valB であり，valA は自社の効率性が上昇するとき（$\tau_n = 1$），valB は自社の効率性が変わらないとき（$\tau_n = 0$）の価値である。

（125）式を計算する準備として locmask を構成する。mask と$2^{\texttt{nfirms}-1}$次元のゼロベクトルを連結した行列が locmask である。ゼロベクトルを連結する位置は locmask の第 place 行である。したがって，locmask は，τ_n を 0 と置いた上で，列ごとに$(\tau_1, \cdots, \tau_{n-1}, \tau_{n+1}, \cdots, \tau_{\texttt{nfirms}})$のあらゆるパターンを並べた，$\texttt{nfirms}\times 2^{\texttt{nfirms}-1}$行列である。行番号は企業のインデックスに対応する。locmask の要素の取る値は 0 か 1 であり，第 j 行が 1 であることは，j 番目の企業の効率性が上昇した（$\tau_j = 1$）ことを意味する。locmask をビット反転した1 – locmask は意味が逆になり，第 j 行が 1 であることは，j 番目の企業の効率性が変わらない（$\tau_j = 0$）ことを意味する。

07，08行目：自社が 1 番目の企業のとき，

$$\texttt{locmask} = \begin{pmatrix} 0\cdots 0 \\ \texttt{mask} \end{pmatrix}. \tag{126}$$

09，10行目：自社が nfirms 番目の企業のとき，

$$\texttt{locmask} = \begin{pmatrix} \texttt{mask} \\ 0\cdots 0 \end{pmatrix}. \tag{127}$$

---

59) 関数 calcval が計算する値は，(77)式で定義される Calcval とは異なる。関数 calcval から参入の有無に応じて２つの将来価値を計算し，期待値を取ったものが Calcval である。Calcval は関数 optimize で計算される。

11～13行目：それ以外では，

$$\texttt{locmask} = \begin{pmatrix} \texttt{mask}\text{の第1行目}\sim\text{第}\texttt{place}-1\text{行目} \\ 0 \quad \cdots\cdots \quad 0 \\ \texttt{mask}\text{の第}\texttt{place}\text{行目}\sim\text{第}\texttt{nfirms}-1\text{行目} \end{pmatrix}. \tag{128}$$

15, 16行目：産業に 1 社しか存在しないときは locmask を使用しないので，その値を 0 とする。

19行目：自社の投資水準を 0 とする。このように設定すると，30行目の計算過程で自社の確率にマスク処理が施される。つまり，23行目より $\texttt{p\_up}(n) = 0$，また $\texttt{locmask}(n, \texttt{i}) = 0$であるから，

$$\texttt{locmask}(n, \texttt{i}).*\texttt{p\_up}(n) + (1 - \texttt{locmask}(n, \texttt{i})).*(1 - \texttt{p\_up}(n)) = 1 \tag{129}$$

となる。

20行目：w の第 place 要素に効率性 k を代入する。

21，22行目：justone は，自社のインデックスに対応する行番号の要素を 1 として，残りの要素をすべて 0 とする，自社のインデックスに目印をつけるベクトルである。つまり，

$$\texttt{justone} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \leftarrow \texttt{place}\text{番目}. \tag{130}$$

23行目：(67) 式に従って，各企業の効率性が上昇する確率を計算する。p_up は nfirms 次元ベクトルである。

次に（125）式を計算する。右辺の V として oldvalue を用いる。まず，

$$\texttt{probmask} = \Pr(\tau_1) \times\cdots\times \Pr(\tau_{n-1}) \times \Pr(\tau_{n+1}) \times\cdots\times \Pr(\tau_{\texttt{nfirms}}) \tag{131}$$

を計算する。この積を計算するために，自社の確率 $\Pr(\tau_n)$にマスク処理をする。さらには，他社を$\tau_j = 0$となる企業と$\tau_j = 1$となる企業とに分けて，前者の企業には

$\Pr(\tau_j = 0)$を，後者の企業には $\Pr(\tau_j = 1)$を対応させて，すべての確率をかけ合わせる。

30行目：locmask の第 i 列は$(\tau_1, \cdots, \tau_{n-1}, 0, \tau_{n+1}, \cdots, \tau_{\mathrm{nfirms}})$（$\tau_j = 0, 1$）となっているとする。locmask(:, i)と p_up を要素ごとにかけることで，$\tau_j = 1$ならば第 j 要素が $\Pr(\tau_j = 1)$, $\tau_j = 0$ならば第 j 要素が 0 となるベクトルが形成される。同様に，1 − locmask(:, i)と 1 − p_up を要素ごとにかけることで，$\tau_j = 0$ならば第 j 要素が $\Pr(\tau_j = 0)$, $\tau_j = 1$ならば第 j 要素が 0 となるベクトルが形成される。したがって，

$$\begin{aligned}&\mathtt{locmask(:, i).*p\_up} + (1 - \mathtt{locmask(:, i)}).*(1 - \mathtt{p\_up})\\ &\quad = 2\mathtt{locmask(:, i)} \times \mathtt{p\_up} + 1 - \mathtt{locmask(:, i)} - \mathtt{p\_up}\\ &\quad = [\Pr(\tau_1), \cdots, \Pr(\tau_{n-1}), 1, \Pr(\tau_{n+1}), \cdots, \Pr(\tau_{\mathrm{nfirms}})]'. \qquad (132)\end{aligned}$$

このベクトルの要素をかけ合わせたものが probmask である。第 n 要素が1となるのは（129）式による。

32行目：locmask のある列について，要素が 1 となる行番号に等しいインデックスを持つ企業は，その効率性が上昇している。したがって，w + locmask は他社の効率性が上昇した後の産業構造であり，これを d とする。

33行目：行列[d, justone]の第２列について，要素が 1 となる行番号が自社のインデックスである。この行列を d について降順に並べ変えたものを temp とする。temp の第２列は justone を並べ替えたものなので，要素が 1 となる行番号を探すことで，自社の新たなインデックスが判明する。

34行目：d に temp の第１列を代入する。

35行目：d のすべての要素から 1 を引く。

36行目：e の要素の中で–1となるものがあれば，0 と設定する。これを e とする。e は$\nu = 1$が実現したときの産業構造である。

37行目：d の要素の中で kmax + 1となるものがあれば，kmax と設定する。d は，$\nu = 0$が実現したときの産業構造となる。

38行目：temp の第２列の要素の中で最大の値は1であり，その要素がある行番号 pl1が自社のインデックスとなる。

40, 41行目：他社の効率性上昇が$(\tau_1, \cdots, \tau_{n-1}, \tau_{n+1}, \cdots, \tau_{\mathtt{nfirms}})$というパターンとなっているとき，

$$\{(1-\mathtt{delta})\times \mathtt{oldvalue}(\mathtt{qencode}(\mathtt{d}), \mathtt{pl1}) + \mathtt{delta}\times \mathtt{oldvalue}(\mathtt{qencode}(\mathtt{e}), \mathtt{pl1})\}\times \mathtt{probmask}$$
$$= \left\{\sum_{\nu=0}^{1} \Pr(\nu) V(\boldsymbol{\omega}+\boldsymbol{\tau}-\nu\boldsymbol{i}, n)\right\} \times \Pr(\tau_1)\times\cdots\times\Pr(\tau_{n-1})\times\Pr(\tau_{n+1})\times\cdots\times\Pr(\tau_{\mathtt{nfirms}}) \qquad (133)$$

を計算する。$\nu = 0$となるときの産業構造はdであり，確率1 − deltaで実現する。他方，$\nu = 1$となるときの産業構造はeであり，確率deltaで実現する。それぞれの産業構造で生み出されるoldvalueを求めてから，νについて期待値を取る。さらにprobmaskをかけて，$(\tau_1, \cdots, \tau_{n-1}, 0, \tau_{n+1}, \cdots, \tau_{\mathtt{nfirms}})$について期待値を取る。**while**ループでは，他社の効率性上昇に関するすべてのパターンi = 1, …, $2^{\mathtt{nfirms}-1}$についてこの値を計算し，valBに足し合わせる。こうして計算されたものが（125）式である。

43行目：w + locmaskにjustoneを加えたものは，自社の効率性が上昇したときの産業構造である。この産業構造をdとする。

44〜52行目：valBの計算と同じ手順でvalAを求める。

5.11　関数encode

```
01   function [out] = encode(ntuple)
02     global binom nfirms
03     code = 1;
04     i = 1;
05     while i <= nfirms
06       digit = ntuple(i);
07       code = code + binom(digit+nfirms+1-i, digit+1);
08       i = i+1;
09     end
10
11     out = code;
12   end
```

エンコード関数は，産業構造ベクトルntupleをエンコードして，1～wmaxの値を取る自然数codeを出力する。関数encodeは二項係数行列Binomを用いてエンコードするので，binomをグローバル変数として参照する。ω_j = ntuple(j)であることから，(58) 式に従って，

$$\begin{aligned}\mathtt{code} = 1 &+ \mathtt{binom}(\mathtt{ntuple}(1) + \mathtt{nfirms}, \mathtt{ntuple}(1) + 1)\\ &+ \mathtt{binom}(\mathtt{ntuple}(2) + \mathtt{nfirms} - 1, \mathtt{ntuple}(2) + 1)\\ &\cdots\cdots\\ &+ \mathtt{binom}(\mathtt{ntuple}(\mathtt{nfirms}) + 1, \mathtt{ntuple}(\mathtt{nfirms}) + 1) \quad (134)\end{aligned}$$

と計算する。

5.12　関数qencode

```
01   function [out] = qencode(ntuple)
02     global encfirm etable multfac binom nfirms
03     if nfirms <= encfirm
04       out = etable(sum(ntuple.*multfac)+1);
05     else
06       out = encode(ntuple);
07     end
08   end
```

クイック･エンコード関数に産業構造ベクトル ntuple を入力すると，エンコードされた産業構造が出力される。まず，ntuple と multfac の内積 i = sum(ntuple.*multfac)を計算して[60]，ntuple に割り振る番号 i = 0, 1, …, (kmax + 1)$^{\mathtt{nfirms}}$ − 1を定める。etable の第 i + 1要素を参照すると，エンコードされた産業構造を得る。

etable を使用するのは，nfirms ≤ encfirm のときだけである。nfirms >

60) MATLAB®では ntuple*multfac'で内積が計算される。

encfirm のときは（58）式に従い，二項係数行列 Binom から直接エンコーディングを行う[61]。

5.13　関数qdecode

```
01    function [out] = qdecode(code)
02      global dtable
03      out = dtable(:, code);
04    end
```

クイック･デコード関数にエンコードされた産業構造 code を入力すると，テーブル・ルックアップ dtable の第 code 列を参照して，デコードされた産業構造ベクトルを出力する。

5.14　関数 decode

```
01    function [out1] = decode(code)
02      global binom nfirms
03      code = code-1;
04      ntuple = zeros(nfirms, 1);
05
06      i = 1;
07      while i <= nfirms;
08        digit = 0;
09        while binom(digit+nfirms-i+2, digit+2) <= code
10          digit = digit+1;
11        end
12        ntuple(i) = digit;
13        code = code - binom(digit+nfirms-i+1, digit+1);
```

61) Goettler のコードに従う。

```
14        i = i+1;
15      end
16
17      out1 = ntuple;
18    end
```

デコード関数は，エンコードされた産業構造 code を，元の産業構造ベクトル ntuple に戻す。3.2節で示した方法に沿って，次のように ntuple(i)（i = 1, …, nfirms）を求める。まず code から 1 を引いたものを，改めて code とする（03行目）。i = 1について，digit = 0として（08行目），

$$\mathtt{binom}(\mathtt{nfirms}+1, 2) \leq \mathtt{code} \tag{135}$$

ならば digit = 1として，binom(nfirms + 1, 2)の斜め右下にある Binom の要素 binom(nfirms + 2, 3)を参照し，code と大きさを比較する。

$$\mathtt{binom}(\mathtt{nfirms}+2, 3) \leq \mathtt{code} \tag{136}$$

ならばさらに digit = 2として，binom(nfirms + 2, 3)の斜め右下にある Binom の要素 binom(nfirms + 3, 4)を参照し，code と大きさを比較する。このように，参照する Binom の要素が code 以下の値である限り，digit に 1 を足し続ける（09～11行目）。digit = a_1のときに初めて，

$$\mathtt{binom}(a_1+\mathtt{nfirms}+1, a_1+2) > \mathtt{code} \tag{137}$$

となったら，ntuple(1) = a_1と確定する（12行目）。つまり，

$$\mathtt{binom}(\mathtt{digit}+\mathtt{nfirm}, \mathtt{digit}+1) \leq \mathtt{code} \tag{138}$$

を満たす最大の digit が a_1である（(59) 式参照）。また code を，

$$\mathtt{code} \leftarrow \mathtt{code} - \mathtt{binom}(a_1+\mathtt{nfirms}, a_1+1) \tag{139}$$

と更新して（13行目），次のループに進む（14行目）。

次に i = 2について，digit = 0と初期化して（08行目），

$$\texttt{binom}(\texttt{nfirms}, 2) \leq \texttt{code} \tag{140}$$

ならば digit に 1 を足して，digit = 1とする。

$$\texttt{binom}(\texttt{nfirms} + 1, 3) \leq \texttt{code} - \texttt{binom}(a_1 + \texttt{nfirms}, a_1 + 1) \tag{141}$$

ならばさらに digit = 2とする。**while** ループの条件が満たされる限りこのプロセスを繰り返して，digit = a_2のときに初めて，

$$\texttt{binom}(a_2 + \texttt{nfirms}, a_2 + 2) > \texttt{code} - \texttt{binom}(a_1 + \texttt{nfirms}, a_1 + 1) \tag{142}$$

となったら，ntuple(2) = a_2と確定する（12行目）。つまり，

$$\texttt{binom}(\texttt{digit} + \texttt{nfirm} - 1, \texttt{digit} + 1) \leq \texttt{code} \tag{143}$$

を満たす最大の digit が a_2である。また code を，

$$\begin{aligned}\texttt{code} \leftarrow \texttt{code} &- \texttt{binom}(a_1 + \texttt{nfirms}, a_1 + 1) \\ &- \texttt{binom}(a_2 + \texttt{nfirms} - 1, a_2 + 1)\end{aligned} \tag{144}$$

と更新して，i = 3に進む。

一般に，引数 code に対して，

$$\begin{aligned}\texttt{binom}(a_i + \texttt{nfirms} - i + 2, a_i + 2) > \texttt{code} &- \texttt{binom}(a_1 + \texttt{nfirms}, a_1 + 1) \\ &- \texttt{binom}(a_2 + \texttt{nfirms} - 1, a_2 + 1) \\ &\cdots\cdots \\ &- \texttt{binom}(a_{i-1} + \texttt{nfirms} - i + 2, a_{i-1} + 1)\end{aligned} \tag{145}$$

を満たす a_iを求めて ntuple(i) = a_iとする。こうして導かれた($a_1, \cdots, a_{\texttt{nfirms}}$)がデコードされた産業構造である。

（続く）

参考文献

[1] 伊里正夫，藤野和健(1985)『数値計算の常識』，共立出版。
[2] 皆本晃弥(2005)『C 言語による数値計算入門：解法・アルゴリズム・プログラム』，サイエンス社。
[3] Brandimarte, P. (2006), *Numerical Methods in Finance and Economics* : *A MATLAB®-Based Introduction*, Wiley.
[4] Doraszelski, U. and A. Pakes. (2007), "A Framework for Applied Dynamic Analysis in IO", In Armstrong, M. and R.Porter.(eds.), *Handbook of Industrial Organization* : *Volume 3*, North-Holland, pp.1887-1966.
[5] Judd, K. (1998), *Numerical Methods in Economics*, MIT Press.
[6] Mathews, J. and K. Fink. (2003), *Numerical Methods Using MATLAB*, Pearson Prentice-Hall.
[7] Miranda, M. and P. Fackler. (2002), *Applied Computational Economics and Finance*, MIT Press.
[8] Pakes, A., G. Gowrisankaran., and P. McGuire. (1993), "Implementing the Pakes-McGuire Algorithm for Computing Markov Perfect Equilibria in Gauss", Working Paper, Yale University.
[9] Pakes, A. and P. McGuire. (1993), "Computing Markov-Perfect Nash Equilibria : Numerical Implications of a Dynamic Differentiated Product Model.", *RAND Journal of Economics*, Vol.25, pp.555-589.
[10] Press, W., S. Teukolsky., W. Vetterling., and B.Flannery.(2007), *Numerical Recpies* : *The Art of Scienctific Computing*, Cambridge University Press.