# 産業動学に関する研究ノート（数値計算編）（終）

加　藤　　浩

## ８．アルゴリズムの改良

以上で解説した Pakes&McGuire アルゴリズムのコードは，アルゴリズムの基本構造に対して忠実にコーディングされたものであるため，収束しやすくしたり，計算量を軽減したりすることを考慮に入れて記述されているわけではない。そのため，アルゴリズムを改良して，より複雑な応用モデルにも耐え得る実用的なコードを構築することが要求される。この節では，Pakes&McGuire アルゴリズムを改良した２つのアルゴリズムを，コードと併せて紹介する。

まず初めに検討するアルゴリズムは，収束の問題に対処するものである。Pakes&McGuire アルゴリズムは，反復を繰り返すことで推測値を改良し，マルコフ完全ナッシュ均衡に収束させることを目指している。指定された最大反復回数のうちに，改良された推測値と前回の反復で計算された推測値との差が，許容誤差の範囲内に収まらないならば収束が失敗したと見なす。Pakes&McGuire アルゴリズムで均衡の推測値を計算してもうまく収束しないときは，代替的な計算手法である FOC 法を用いて推測値を計算することで，収束が成功することを期待する（Goettler アルゴリズム）。

次に検討するアルゴリズムは，計算量を軽減して，計算時間を短縮するアルゴリズムである[87), 88)]。理論上はアルゴリズムの収束が保証されていたとしても，１

87) MATLAB®固有の高速化として，Parallel Computing Toolbox を利用した並列計算がある。関数 `contract` における産業構造のループについて，

```
parfor w = 1: wmax
```

とすることで，各産業構造の処理を各 CPU コアに割り当てて並行処理を行う。

反復当たりに必要となる計算時間が長くなれば，それだけ収束するまでに膨大な時間を要する。計算時間の多くが，将来価値(125)式を計算することに費やされる[89]。将来価値の計算とは，次期に実現しうる産業構造$\boldsymbol{\omega}'$のすべてについて価値$V(\boldsymbol{\omega}', n)$を計算して，それに$\boldsymbol{\omega}'$が実現する確率をかけ合わせて期待値を取ることである。企業数$N$が増えると，実現しうる産業構造の数が指数関数的に増え，期待値の計算時間が爆発的に増大する。これが次元の呪いと呼ばれる現象である。そこでモデルを連続時間で構築することで，次元の呪いを回避することを試みる。連続時間モデルでは，次の時点で起こり得る産業構造の数が激減するので，期待値の計算量が減り，処理の高速化が期待できる（Doraszelski&Judd アルゴリズム）。

### 8.1　FOC 法

Pakes&McGuire アルゴリズムは，均衡投資水準を反復最適反応法（iterated best response algorithm）により求めている。$i$ 回目の反復で導出される企業 $n$ の投資水準 $x_n^{(i)}$ は，他社の投資水準 $(x_1^{(i)},\cdots,x_{n-1}^{(i)},x_{n+1}^{(i-1)},\cdots,x_N^{(i-1)})$ に対して，企業 $n$ の期待割引現在価値を最大にするものであり，このような意味で最適反応となっている（脚注 50) 参照)。すべての企業について最適反応を計算する関数が `optimize` であり，$\boldsymbol{x}^{(i-1)}$=`oldx` を $\boldsymbol{x}^{(i)}$=`newx` に写す関数である[90]。ここで，$\boldsymbol{x}^{(i)}=(x_1^{(i)},\cdots,x_N^{(i)})$ は，$i$ 回目の反復で導き出される投資ベクトルである。この関数を $\boldsymbol{F}$ で表すと，$\boldsymbol{x}^{(i)}=\boldsymbol{F}(\boldsymbol{x}^{(i-1)})$と表すことができる。$\boldsymbol{x}^{(i)}$が収束するまで反復を繰り返し，収束した点がマルコフ完全ナッシュ均衡になる。このことから，反復最適反応法はクールノーの調整過程と同じ処理を行っている。

---

88) 計算時間は，「1 状態当たりに必要な計算時間×1 反復当たりの状態数×収束に必要な反復回数」で決まる。産業動学モデルの状態は$(\boldsymbol{\omega}, n)$，$\boldsymbol{\omega}\in\Omega$，$n=1,\cdots,N$ である。したがって，市場に収容できる最大企業数 $N$ や効率性が取る値の数 $\overline{\omega}$ が増えると，実現可能な状態数が指数関数的に増加する。(5) 式のように産業構造集合に制限を課すと，指数関数的な増加を回避することができる。

89) 将来価値を計算する関数 `calcval` は計算集約的であるので，C 言語でコーディングし，これを MEX ファイルに変換して MATLAB®から呼び出すことで，計算を高速化することができる。

90) (77)，(79)式で定義される関数である。他社の投資水準は，効率性上昇の確率を通じて自社の価値に影響を与える（(77)式)。

これに対して Goettler アルゴリズムは，FOC 法により均衡投資水準を計算している。任意の期における任意の産業構造の下で $N$ 社が投資水準を決定する静学ゲームを考え，このゲームのナッシュ均衡を計算する。投資ベクトル $\boldsymbol{x}$ は次期のゲームの状態$(\boldsymbol{\omega}', n)$が実現する確率に影響を与えるが，次期以降で展開されるゲームの結果は継続価値の期待値 $\mathrm{E}(V(\boldsymbol{\omega}', n)|\, \boldsymbol{x})$に集約される。将来のどの期においても期待割引現在価値を最大にするように投資水準が選択されており，最大化された期待割引現在価値が $\mathrm{E}(V(\boldsymbol{\omega}', n)|\, \boldsymbol{x})$になる。したがって，FOC 法は逆向きの推論により投資水準を求めていることと同じである。

ナッシュ均衡投資水準 $\boldsymbol{x}^*$は $\boldsymbol{x}^* = \boldsymbol{F}(\boldsymbol{x}^*)$を満たすので，FOC 法は非線形連立方程式 $\boldsymbol{x} - \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{o}$ の解を求めることに帰着される。ナッシュ均衡における企業 $n$ の投資水準を $x(\boldsymbol{\omega}, n)$とすると，均衡価値は $V(\boldsymbol{\omega}, n) = \pi(\boldsymbol{\omega}, n) - x(\boldsymbol{\omega}, n) + \beta \mathrm{E}(V(\boldsymbol{\omega}', n)|\, \boldsymbol{x})$によって計算される。右辺の $V$ は，前回の反復で計算された推測値 `oldvalue` を用いる。こうして計算された均衡価値を `newvalue` とする。`newvalue` と `oldvalue` の差が，許容誤差の範囲内に収まるまで反復を繰り返す。Goettler アルゴリズムのメイン・ループとして，5.1節のコードをそのまま使用することができるが，そこから呼び出される関数は改良を要する[91]。

### 8.1.1　関数 contract

```
01    function [] = contract()
02      global wmax nfirms oldx newvalue newx foc_tol
03
04      for w = 1: wmax
05        locw = qdecode(w);
06
07        do = (1:nfirms)';
08        do = do(locw ~= ([-1; locw(1:nfirms-1)]) & locw>0);
```

91) 関数 `update` は使用しない。産業に収容できる最大企業数を 1 社ずつ増やして，そのたびマルコフ完全ナッシュ均衡を計算するのではなく，最初から `nfirms` 社問題を考えて均衡を計算する。

```
09
10        [tmp] = newton(oldx, 'get_foc', w, do);
11        oldx(w, do) = tmp';
12
13        get_foc(oldx, w);
14      end
15    end
```

産業構造 w について巡回して，各々の w に対して関数 newton と関数 get_foc を呼び出す。newton で均衡投資水準を，get_foc で均衡価値をそれぞれ計算する[92]。企業 n についての巡回は関数 get_foc の中で行われる。

05行目：産業構造wをデコードして産業構造ベクトルlocw = (locw(1), …, locw(nfirms))を得る。

07，08行目：同じ効率性を持つ企業は同じ投資水準を選択するという対称性の制約を課す（脚注 12) 参照)。次のようにして効率性の異なる企業のインデックスを取り出して，インデックス・ベクトルdoに格納する。locwの要素を１つ右にずらしたベクトル(–1, locw(1), …, locw(nfirms – 1))を考える。このベクトルとlocwを要素ごとに比較して異なる値を持つとき，企業nと企業n + 1の効率性が異なる。このようなインデックスnをdoに順次格納する。doに含まれるインデックスを持つ企業について，ニュートン＝ラフソン法を用いて均衡投資水準を計算する。doに含まれないインデックスを持つ企業の投資水準については，関数get_focにおいて対称性の制約を考慮した処理がなされる。

10行目：関数newtonに初期点oldx(w, do)を入力して，foc = get_foc($x^*$, w) = 0となる解$x^*$をニュートン＝ラフソン法により求める。get_focはoldxを入力す

92) オリジナルの Goettler アルゴリズムでは，投資水準，退出確率，参入確率をセットにした政策関数ベクトル x_exit_entry = (oldx, oldexit, oldentry)に対して反復を行っている。関数 get_foc で x_exit_entry を更新して(newx, newexit, isentry)を計算した上で，foc = (oldx, oldexit, oldentry) – (newx, newexit, isentry)を返す。isentry は関数 chkentry で計算され，newexit は関数 optimize で計算される。いずれの関数も get_foc から呼び出される。

ると，その最適反応newxを計算し，foc = oldx − newxを返す関数である。したがって，get_focの根を求めることは，oldx = newxとなる投資水準oldxを求めることを意味する[93]。

11行目：newtonで求めた根tmpをoldx(w, do)に上書きする。

13行目：newtonで求めた根oldx(w, do)を関数get_focに入力して，均衡価値の新しい推測値newvalue(w, do)を得る。同時に均衡投資水準の新しい推測値newx(w, do)も計算されるが，これはoldx(w, do)と同じ値になっている。

### 8.1.2 関数 newton

```
01   function [tmp] = newton(x, objfunk, w, do)
02     global foc_tol nfirms
03     p = x(w, do)';
04     np = size(p, 1);
05     lb = zeros(nfirms, 1);
06     ub = ones(nfirms, 1) + 99999999;
07     deriv = zeros(np, np);
08     iter = 0;
09
10     while iter < 4000*np
11       iter = iter+1;
12
13       [y] = feval(objfunk, x, w);
14       z = y(do);
15
16       if max(abs(z)) < foc_tol
17         break;
18       end
19
```

93) 反応関数 $F(x)$の不動点を求めることと同じである。

```
20      for i = 1: np
21        dx = x;
22
23        if ub(do(i)) - p(i) < 0.0001
24          dp(i) = p(i) - 0.0001;
25        else
26          dp(i) = p(i) + 0.0001;
27        end
28
29        dx(w, do(i)) = dp(i);
30        [dy] = feval(objfunk, dx, w);
31        dz = dy(do);
32
33        deriv(:, i) = (dz - z)/(dp(i) - p(i));
34      end
35
36      if rcond(deriv) < 1e-14
37        pnew = lb(do) + ...
38          rand(np, 1).*(min([ub(do)'; 10*ones(1, np)])' - lb(do));
39      else
40        pnew = deriv¥z;
41      end
42
43      if mod(iter, 200*np) < 5
44        pnew = p - pnew*((0.1 + (mod(iter, 200*np))/10)*rand(1, 1));
45      else
46        pnew = p - pnew;
47      end
48
49      pnew = min([ub(do)';  pnew'])';
```

```
50        pnew = max([lb(do)';  pnew'])';
51        p  = pnew;
52        x(w, do) = p';
53      end
54
55      tmp = x(w, do)';
56    end
```

x(w, do)を初期点として，ニュートン＝ラフソン法を用いて関数 objfunk の根を求める。ただし，do に含まれるインデックスを持つ要素についてのみ根を計算する。

03，04行目：x(w, do)の値を p とする。つまり，p(i) = x(w, do(i))である。p は np 次元ベクトルであり，do と同じ次元である。この p を根の推測値として，反復を繰り返して更新していく。

05，06行目：根の下限 lb，上限 ub を設定する。ここでの設定は，上限が実質存在せず，0 よりも大きい根を求めるものとする。

07，08行目：偏微分係数行列 deriv，および反復回数 iter を初期化する。

10行目：収束に必要な反復回数は根の次元 np に比例するので，最大反復回数を 4000np とする。

11行目：反復回数を 1 つ増やす。

13行目：y = objfunk(x, w)とする。

14行目：y の要素のうち，do が指し示すインデックスの要素だけ取り出して z とする。

16行目：z の値がすべて 0 になるとき，z は objfunk の根となる。そこで，根の判定条件を次のように設定する。z の要素の絶対値のうち，最大値が許容誤差 foc_tol よりも小さいならば[94]，根を見つけたものとして **while** ループから抜け出す。

94) foc_tol = $10^{-7}$ としている。

20～34行目で objfunk の偏微分係数行列を計算する。

20行目：p の要素(p(1), …, p(np))についてループする。

21行目：p(i－1)番目のループで dx(w, do(i－1)) = p(i－1) + $\varepsilon$としていたものを，p(i)番目のループでは dx(w, do(i－1)) = x(w, do(i－1))と元の値に戻す。

23, 24行目：$\varepsilon$ = 0.0001とする。点 p(i)と上限 ub(do(i))の距離が$\varepsilon$未満であるとき，p(i)を$\varepsilon$だけ減らした値を dp(i)とする[95]。

25，26行目：p(i)と ub(i)の距離が$\varepsilon$以上であるとき，p(i)を$\varepsilon$だけ増やした値を dp(i)とする。

29行目：dp(i)を dx の第 w 行，第 do(i)列に代入する。したがって，dx(w, do(i)) = (x(w, 1), …, x(w, do(i) – 1), p(i) + $\varepsilon$, x(w, do(i) + 1), …, x(w, nfirm))となる。

30行目：dy = objfunk(dx, w)とする。

31行目：dy の要素のうち，do が指し示すインデックスの要素だけ取り出して dz とする。つまり，dz = (objfunk(dx(w, do(1)), w), …, objfunk(dx(w, do(np)), w))，また，dz(i) = objfunk(x(w, 1), …, x(w, do(i) – 1), p(i) + $\varepsilon$, x(w, do(i) + 1), …, x(w, nfirm), w)となる。

33行目：点 p(i)における objfunk の偏微分係数の近似値を計算して（(194)式参照），deriv の第 i 列に格納する（(191)式参照）。

36～38行目：deriv の条件数の逆数が極めて小さいとき，連立方程式 deriv×pnew = z の解 pnew は，z の丸め誤差に対して感応度が高くなる。したがって，たとえ z の丸め誤差が小さくても，解 pnew の誤差は大きくなり，この解が次の反復では連立方程式の右辺に置き換えられるので，解 pnew の誤差はさらに拡大する。また，収束するスピードも遅くなる。そのような理由で，悪条件である deriv を使ってステップを計算するのを止めて，その代わりに[0, 1]上の一様乱数を用いてステップを設定する[96]。

39，40 行目：deriv が良条件であるときは，ニュートン・ステップ–objfunk(x, w) · (∂objfunk(x, w) / ∂x)$^{-1}$ を使用する。

---

95) ここでは上限を非常に大きな値に設定しているので，この条件式は実行されない。

96) ここでは，上限と下限の設定から，pnew = 10×rand をステップとする

ニュートン・ステップを計算したので，(193)式に従って，根の推測値 p = x(w, do)をニュートン・ステップだけ進めて，改良された根の推測値 pnew を導き出す。ただし，収束しない事態を避けるために，減衰ニュートン＝ラフソン法を適時実行する。

43〜47行目：反復回数が200np の倍数であるときは，ニュートン・ステップに減衰係数0.1を乗じて，ニュートン・ステップよりも少し勾配を緩めてステップを進める。続く 4 回の反復でも減衰ニュートン＝ラフソン法を実行するが，回数ごとに減衰係数を変える。すなわち，$r$ を反復回数を200np で割ったときの剰余，rand を[0, 1]上の一様乱数すると，減衰係数を$(1 + r \times \text{rand}) \div 10$と置く（44行目）。それ以外の反復では，通常のニュートン・ステップで根の推測値を改良する（46行目）。

49〜51行目：pnew が[lb, ub]に収まる値なら pnew を，そうでないならば lb，あるいは ub を改めて pnew とする。この pnew を p に上書きして，再度ニュートン＝ラフソン法を実行する。

52行目：根の推測値 p(i)を x の第 w 行，第 do(i)列に代入する。

55行目：収束した根の推測値 x(w, do)を戻り値とする。

### 8.1.3　関数 get_foc[97)]

```
01    function [foc] = get_foc(x, w)
02      global nfirms newvalue newx
03      y = zeros(1, nfirms);
04      y = x(w, :);
05      locw = qdecode(w)';
06
07      for i = 2:nfirms
08        if locw(i) == locw(i-1);
09          y(i) = y(i-1);
10        end
11      end
```

97) 反復最適反応法は関数 get_foc を 1 回だけ呼び出すことに等しい。

```
12
13      chkentry;
14      [newx(w, :), newvalue(w, :)] = optimize(w);
15
16      foc = y - newx(w, :);
17    end
```

投資水準 x と産業構造 w を引数に取り，産業構造 w における x の最適反応 newx を計算する。そして，x と newx の差を戻り値とする。

03行目：y を初期化する。

04行目：入力された産業構造 w における各企業の投資水準 x を y に代入する。

05行目：産業構造 w をデコードして産業構造ベクトル locw を得る。

07〜11行目：対称性の制約により，同じ効率性を持つ企業の投資を同じ水準に設定する。

13行目：関数 chkentry を呼び出して参入確率 isentry を計算する[98]。

14行目：関数optimizeを呼び出し，産業構造wにおける均衡価値newvalueと均衡投資水準newxを計算する。

16行目：xとnewxの差focを計算する。

### 8.1.4　関数 optimize

oldvalue，oldx を基にして，改良された新しい推測値 newvalue，newx を計算する。基本的な処理は 5.8 節と同じであるが，Goettler アルゴリズムでは退出確率を計算している点が，Pakes&McGuire アルゴリズムと異なる[99]。つまり，

98) Pakes&McGuire アルゴリズムでは，関数 contract の中で関数 chkentry を 1 回だけ呼び出しているが，Goettler アルゴリズムでは，産業構造 w ごとに関数 chkentry を呼び出して参入確率を更新する。したがって，5.9 節の関数 chkentry から **while** w = 1 : wmax の記述を削除する。

99) Pakes&McGuire アルゴリズムでは，入力された oldvalue に対して退出する企業を決定し，退出が完了した後の産業構造の下での参入確率を計算して，参入の可能性を考慮に入れた将来価値から投資水準を計算する。Goettler アルゴリズムでは，関数 get_foc で投資水準，退出確率，参入確率が同時に計算される（脚注 92）参照）。

Pakes&McGuire アルゴリズムでは，スクラップ価値を確定変数として扱っているが，Goettler アルゴリズムでは，スクラップ価値を確率変数としている。ここでは，スクラップ価値$\phi$を$[\phi_L, \phi_H]$上の一様分布に従い，企業間で独立に分布する確率変数とする。既存企業はスクラップ価値$\phi$の実現値を知った後に退出するか事業を継続するかを選択する。他社のスクラップ価値は観察不可能であり，その意味で確率変数となるので，他社による退出政策の決定は退出確率の決定と同義である[100), 101)]。

継続価値$V(\boldsymbol{\omega}, n)$と実現したスクラップ価値$\phi$とを比較して，事業を継続するか退出するかを決める[102)]。$V(\boldsymbol{\omega}, n) < \phi_L$のときは，スクラップ価値の実現値がどのようなものであっても退出を選択するので，退出確率は1となる。このとき，退出価値はスクラップ価値の期待値に等しく，

$$\mathrm{E}(\phi) = \frac{\phi_L + \phi_H}{2} \tag{198}$$

となる。$V(\boldsymbol{\omega}, n) \in [\phi_L, \phi_H]$のときは，スクラップ価値が$\phi \geq V(\boldsymbol{\omega}, n)$ならば退出する。したがって退出確率は，

$$r(\boldsymbol{\omega}, n) = \frac{\phi_H - V(\boldsymbol{\omega}, n)}{\phi_H - \phi_L} \tag{199}$$

となり，退出価値の期待値は，

---

100) 投資政策に加えて，退出政策や参入政策も連続変数となる。関数 `optimize` は (`oldx`, `oldexit`, `oldentry`)を(`newx`, `newexit`, `isentry`)に写す反応関数であり，FOC 法はこの反応関数の不動点を計算する。価値関数は連続関数であるから，反応関数もまた連続となる。したがって，ブラウアーの不動点定理から，不動点の存在が保証される。また，政策が連続変数であるとき，反復による収束は成功しやすくなる（脚注 5) 参照)。

101) 実現するスクラップ価値は企業ごとに異なるので，Pakes&McGuire アルゴリズムが前提としている退出政策「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」を破棄する。

102) 効率性が上昇する確率を $p$ とし，$v_1, v_2$ を (83)，(84) 式で定義する。Goettler アルゴリズムでは，退出を決定した期末にも利潤$\pi(\boldsymbol{\omega}, n)$を獲得することから，

$$V(\boldsymbol{\omega}, n) = -x(\boldsymbol{\omega}, n) + \beta\{pv_1 + (1-p)v_2\}$$

とする。Pakes&McGuire アルゴリズムでは，

$$V(\boldsymbol{\omega}, n) = \pi(\boldsymbol{\omega}, n) - x(\boldsymbol{\omega}, n) + \beta\{pv_1 + (1-p)v_2\}$$

としている。

$$\mathrm{E}(\phi \mid \phi > V(\boldsymbol{\omega}, n)) = \frac{\phi_H + V(\boldsymbol{\omega}, n)}{2} \tag{200}$$

となる[103]。

## 8.2 連続時間モデル

### 8.2.1 モデルの設定

企業 $n$ の効率性が $\omega_n$ であるとき，投資の結果により次期の効率性は，$\omega_n' = \omega_n, \omega_n + 1$ のうちのどれかに推移する。また，産業全体ショックが発生すると，すべての企業の効率性が１つ減少する。Pakes&McGuire アルゴリズムでは，１期間内で複数企業の効率性が同時に変化する可能性があることから，自社の効率性を所与として，次期に起こり得る他社の効率性プロファイル $\omega_{-n}$ は $2^N$ 通りある。したがって，自社の効率性を所与として，次期に実現しうる産業構造 $\boldsymbol{\omega}'$ は $2^N$ 通りある。期待割引現在価値を計算するには，これだけの数の和を取る必要があり，計算量は $N$ について指数関数的に増加する。

Doraszelski&Judd アルゴリズムは無限連続時間でモデルを構築している。効率性の瞬時の変化を「ジャンプ」と呼ぶことにすると，ジャンプが発生する要因は次の４つである。ｉ）投資によりある企業の効率性が上昇する，ii）産業全体ショックによりすべての企業の効率性が低下する，iii）退出発生により空きスロットが生じる，iv）参入発生により空きスロットに参入企業の効率性が埋まる。これらのジャンプが一時点で同時に発生することはなく，発生するジャンプはせいぜい１つである。したがって，効率性が変化するパターンは $2N + 2$ 通りあり，それだけの数の産業構造 $\boldsymbol{\omega}'$ について期待値を取れば期待割引現在価値が計算できる[104]。このように，計算量は $N$ について線形的に増加するので，企業数が増え

103) 関数 `calcval` では，投資による効率性上昇，産業全体ショックによる効率性低下に加えて，他社の退出も考慮に入れて継続価値の期待値を計算する。他社についてどの企業が退出するかという退出パターン（$2^{\mathtt{nfirms}-1}$ 通りある）に関するループを考え，それが実現する確率と，そのときの継続価値をかけ合わせて，すべての退出パターンについてそれらを足し合わせて期待値を求める。この期待値を計算する方法は効率性上昇の場合と同じであり，`probmask` に対応するものとして `Exprobmask` を，`locmask` に対応するものとして `EXITmask` を用意して，`p_up` の代わりに確率退出 `oldexit` を使って計算する。

104) (213) 式から分かるように，価値の変化分について期待値を取っているので，ジャンプが発生せず産業構造が変化しないケースは考えなくても良い。

るほど，Pakes&McGuire アルゴリズムと比べて計算量が大幅に減り，それだけ収束するまでに費やされる時間の短縮が期待できる。

$t$ 時点の産業構造が $\boldsymbol{\omega}(t)$ であるとき，ジャンプが発生することで，産業構造は次のように変化する。

$$\boldsymbol{\omega}' = \begin{cases} (\varnothing, \omega_{-n}(t)) \ (n = 1, \cdots, N) & (\text{企業 } n \text{ が退出する}) \\ \boldsymbol{\omega}(t) \cup \omega^e & (\text{参入が発生する}) \\ (\omega_n(t) + 1, \omega_{-n}(t)) \ (n = 1, \cdots, N) & (\text{企業 } n \text{ の効率性が上昇する}) \\ \boldsymbol{\omega}(t) - \boldsymbol{i} & (\text{産業全体ショックが発生する}) \\ \boldsymbol{\omega}(t) & (\text{ジャンプが発生しない}) \end{cases} \tag{201}$$

ここで，$(\varnothing, \omega_{-n}(t))$ は $\boldsymbol{\omega}(t)$ の要素 $\omega_n(t)$ を空きスロットに置き換えたもの，$\boldsymbol{i}$ はすべての要素が 1 であるベクトル，$\boldsymbol{\omega}(t) \cup \omega^e$ はベクトル $\boldsymbol{\omega}(t)$ に要素 $\omega^e$ を付け加えて降順に並べ替えたベクトルである。

効率性のジャンプは，ポアソン過程に従うランダムな時間に発生する[105]。ジャンプ発生のそれぞれの要因に対して，ハザード率を特定化する。

・退出のハザード率について

退出するときは保有する資産を売却する。毎時点，資産の購入者がハザード率 $\eta$ で現れ，購入価格 $\phi$ を既存企業に提示する。$\phi$ は $[\phi_L, \phi_H]$ 上の一様分布からランダムに取り出される確率変数である。購入者が提示した購入価格に対して，既存企業は受け入れるか拒否するかのどちらかを選択する。提示された購入価格が継続価値よりも大きければ，提示を受け入れて資産を売却し退出する。提示を拒否するときは事業を継続する。現在の産業構造が $\boldsymbol{\omega}(t)$ であるとき，継続価値は $V(\boldsymbol{\omega}(t), n)$ となり，実現したスクラップ価値 $\phi$ が $V(\boldsymbol{\omega}(t), n) \leq \phi$ となると資産を売却して退出するので，資産の購入者が現れるという条件の下での退出確率は (199) 式になる。したがって，退出発生のハザード率は $\eta r(\boldsymbol{\omega}(t), n)$ となる。

105) ジャンプする時間がポアソン過程に従うとき，$t$ 時点前にジャンプが発生する確率は $F(t) = 1 - e^{-\lambda t}$ となる。$t$ 時点まではジャンプが発生していない条件の下で，$[t, t + \Delta]$ 時点でジャンプが発生する条件付き確率は，

$$\frac{F'(t)\Delta}{1 - F(t)} = \lambda\Delta$$

となり，単位時間当たりで測った条件付き確率 $\lambda$ がハザード率である。

・参入のハザード率について

参入して事業を行うためには資産を取得しないといけない。毎時点，資産の販売者がハザード率$\eta^e$で現れ，販売価格 $x^e$ を提示する。$x^e$ は$[x_L^e, x_H^e]$上の一様分布からランダムに取り出される確率変数である。販売者が提示した販売価格に対して，参入企業は受け入れるか拒否するかのどちらかを選択する。参入価値が提示された販売価格を上回るならば，提示を受け入れて資産を購入した上で参入する。参入した瞬間に投資をすることは不可能であるとする。

現在の産業構造が$\boldsymbol{\omega}(t)$のとき，参入企業 $n$ が参入するならば，空きスロットに効率性$\omega^e$が埋まり，参入発生後の産業構造は$\boldsymbol{\omega}(t) \cup \omega^e$ となる。参入価値は $V(\boldsymbol{\omega}(t) \cup \omega^e, n)$となり，販売価格 $x^e$ が $V(\boldsymbol{\omega}(t) \cup \omega^e, n) \geq x^e$ となるときに参入する。したがって，資産の販売者が現れるという条件の下での参入確率は，

$$\lambda(\boldsymbol{\omega}(t)) = \frac{V(\boldsymbol{\omega}(t) \cup \omega^e, n) - x_L^e}{x_H^e - x_L^e} \tag{202}$$

となり，参入発生のハザード率は$\eta^e \lambda(\boldsymbol{\omega}(t))$となる。

・投資のハザード率について

$t$ 時点における企業 $n$ の投資水準 $x$ は，次の時点で効率性が$\omega_n(t)$から$\omega_n(t) + 1$ へと上昇する確率を決める。効率性上昇が生じるハザード率を $x^\alpha$とする。ここで，$\alpha$は投資水準に対するハザード率の弾力性であり，一定数である。

・産業全体ショックのハザード率について

産業全体にショックが発生するとき，すべての既存企業の効率性が低下して，産業構造が$\boldsymbol{\omega}(t)$から$\boldsymbol{\omega}(t) - \boldsymbol{i}$ へと変化する。産業全体ショックが発生するハザード率を$\delta$とする。

以上より，現在の産業構造が$\boldsymbol{\omega}(t)$のとき，$\boldsymbol{\omega}'$へのジャンプが発生するハザード率$\mu(\boldsymbol{\omega}' | \boldsymbol{\omega}(t))$は次のようになる。

$$\mu((\varnothing, \omega_{-n}(t)) | \boldsymbol{\omega}(t)) = \eta r(\boldsymbol{\omega}(t), n) \quad (n = 1, \cdots, N), \tag{203}$$

$$\mu(\boldsymbol{\omega}(t) \cup \omega^e | \boldsymbol{\omega}(t)) = \eta^e \lambda(\boldsymbol{\omega}(t)), \tag{204}$$

$$\mu((\omega_n(t)+1,\ \omega_{-n}(t)) \mid \boldsymbol{\omega}(t)) = \{x(\boldsymbol{\omega}(t), n)\}^{\alpha} \quad (n = 1, \cdots, N), \tag{205}$$

$$\mu(\boldsymbol{\omega}(t) - \boldsymbol{i} \mid \boldsymbol{\omega}(t)) = \delta. \tag{206}$$

$\Delta$を短い時間間隔とすると，時間$[t, t+\Delta]$において，２企業以上の効率性が同時にジャンプすることはない。この事実を以下に示す。産業構造が$\boldsymbol{\omega}(t)$であるとき，$[t,\ t+\Delta]$時点における企業 $n$ のハザード率をすべて足し合わせたものを $h_n(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t))$とする[106]。時間$[t, t+\Delta]$でジャンプが発生する確率は，

$$\Pr(\omega_n(t+\Delta) \neq \omega_n(t) \mid \boldsymbol{\omega}(t)) = h_n(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t))\Delta + O(\Delta^2) \tag{207}$$

となるので，企業 $n$ と企業 $m$ にジャンプが同時に発生する確率は，

$$\begin{aligned}
&\Pr(\omega_n(t+\Delta) \neq \omega_n(t) \mid \boldsymbol{\omega}(t)) \times \Pr(\omega_m(t+\Delta) \neq \omega_m(t) \mid \boldsymbol{\omega}(t)) \\
&\quad = h_n(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t)) \times h_m(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t)) \times \Delta^2 \\
&\qquad + \{h_n(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t)) + h_m(\boldsymbol{\omega}(t+\Delta)|\ \boldsymbol{\omega}(t))\}\Delta \times O(\Delta^2) + O(\Delta^2) \times O(\Delta^2) \\
&\quad = O(\Delta^2)
\end{aligned} \tag{208}$$

となり，短い時間間隔 $\Delta$ に対して，この確率は限りなくゼロに近づく。

次に既存企業の最適化問題を考える。割引因子を$\rho$とする[107]。産業構造が$\boldsymbol{\omega}$であるとき，企業 $n$ の瞬時的な利潤フローは$\pi(\boldsymbol{\omega},\ n)$となる。短い時間間隔$\Delta$について，ベルマン方程式は次のように定式化できる。

$$V(\boldsymbol{\omega}, n) = \max_{x}\{\pi(\boldsymbol{\omega}, n) - x\}\Delta + e^{-\rho\Delta}\mathrm{E}\left(\mathrm{Max}\{\chi\phi', V(\boldsymbol{\omega}', n)\} \mid \boldsymbol{\omega}, \boldsymbol{x}\right). \tag{209}$$

$\Delta$時間経過後にスプラック価値の実現値が分かるので，$t$ 時点の段階ではスプラック価値$\phi'$は確率変数となる。また，資産の購入者が現れるときに，$\chi = 1$ となる。$e^{-\rho\Delta} \fallingdotseq 1 - \rho\Delta$と近似できることから，ベルマン方程式は次のように書き換えられる。

$$V(\boldsymbol{\omega}, n) \fallingdotseq \max_{x}\{\pi(\boldsymbol{\omega}, n) - x\}\Delta + (1 - \rho\Delta)\mathrm{E}\left(\mathrm{Max}\{\chi\phi', V(\boldsymbol{\omega}, n)\} \mid \boldsymbol{\omega}, \boldsymbol{x}\right). \tag{210}$$

ハザード率を明示すると，２つのジャンプが同時に発生することはないことから，この式は以下のように書くことができる。

---

106) 同一企業に複数のジャンプが発生することはない。効率性の上昇と下落が同時に起こることは，複数企業の効率性が同時にジャンプすることと同じであるから，この確率はゼロである。また，効率性の上昇と退出が同時に発生することは，退出のみ発生することと同じである。さらに，同一企業に参入と退出が同時に発生することもない。

107) 離散期間モデルの割引因子を$\beta$とすると。$\rho = -\log\beta$となる。

$$V(\boldsymbol{\omega},n) \fallingdotseq \underset{x}{\mathrm{Max}}\{\pi(\boldsymbol{\omega},n)-x\}\Delta + (1-\rho\Delta)\Big[\underbrace{\eta r(\boldsymbol{\omega},n)\Delta \mathrm{E}(\phi \mid \phi \geq V(\boldsymbol{\omega},n))}_{\text{自社の退出}}$$

$$+\underbrace{x^{\alpha}\Delta V((\omega_n+1,\omega_{-n}),n)}_{\text{自社の投資}} + \underbrace{\sum_{m\neq n} x_j{}^{\alpha}\Delta V((\omega_m+1,\omega_{-m}),n)}_{\text{他社の投資}} + \underbrace{\sum_{m\neq n}\eta r(\boldsymbol{\omega},m)\Delta V((\varnothing,\omega_{-m}),n)}_{\text{他社の退出}}$$

$$+\underbrace{\eta^{e}\lambda(\boldsymbol{\omega})\Delta V(\boldsymbol{\omega}\cup\omega^{e},n)}_{\text{参入}} + \underbrace{\delta\Delta V(\boldsymbol{\omega}-\boldsymbol{i},n)}_{\text{産業全体ショック}}\Big]$$

$$+\underbrace{(1-\rho\Delta)\left[1-\left\{x^{\alpha}+\sum_{m\neq n}x_m{}^{\alpha}+\sum_{m=1}^{N}\eta r(\boldsymbol{\omega},m)+\eta^{e}\lambda(\boldsymbol{\omega})+\delta\right\}\Delta\right]}_{\text{ジャンプなし}}V(\boldsymbol{\omega},n). \quad (211)$$

$\Delta\to 0$ とすると

$$\begin{aligned}\rho V(\boldsymbol{\omega},n) \fallingdotseq \underset{x}{\mathrm{Max}}\{&\pi(\boldsymbol{\omega},n)-x\}+\eta r(\boldsymbol{\omega},n)\{\mathrm{E}(\phi\mid\phi\geq V(\boldsymbol{\omega},n))-V(\boldsymbol{\omega},n)\}\\ &+x^{\alpha}\{V((\omega_n+1,\omega_{-n}),n)-V(\boldsymbol{\omega},n)\}\\ &+\sum_{m\neq n}x_m{}^{\alpha}\{V((\omega_m+1,\omega_{-m}),n)-V(\boldsymbol{\omega},n)\}+\sum_{m\neq n}\eta r(\boldsymbol{\omega},m)\{V((\varnothing,\omega_{-m}),n)-V(\boldsymbol{\omega},n)\}\\ &+\eta^{e}\lambda(\boldsymbol{\omega})\{V(\boldsymbol{\omega}\cup\omega^{e},n)-V(\boldsymbol{\omega},n)\}+\delta\{V(\boldsymbol{\omega}-\boldsymbol{i},n)-V(\boldsymbol{\omega},n)\}\end{aligned} \quad (212)$$

となる。企業 $n$ のハザード率をすべて足し合わせたものを $h_n(\boldsymbol{\omega}'\mid\boldsymbol{\omega})$とすると，上の式は次のように書くことができる。

$$\rho V(\boldsymbol{\omega},n) \fallingdotseq \underset{x}{\mathrm{Max}}\{\pi(\boldsymbol{\omega},n)-x\}+\sum_{\boldsymbol{\omega}'}h_n(\boldsymbol{\omega}'\mid\boldsymbol{\omega})\{V(\boldsymbol{\omega}',n)-V(\boldsymbol{\omega},n)\}. \quad (213)$$

ただし表現の便宜上，自社の退出に伴う価値の変化 $\mathrm{E}(\phi\mid\phi\geq V(\boldsymbol{\omega},n))-V(\boldsymbol{\omega},n)$も $V(\boldsymbol{\omega}',n)-V(\boldsymbol{\omega},n)$で表している。

均衡投資水準は，

$$x(\boldsymbol{\omega},n)=\arg\underset{x}{\mathrm{Max}}[-x+x^{\alpha}\{V((\omega_n+1,\omega_{-n}),n)-V(\boldsymbol{\omega},n)\}] \quad (214)$$

によって求められる。したがって，

$$x(\boldsymbol{\omega},n)=\mathrm{Max}\left\{0,\alpha\{V((\omega_n+1,\omega_{-n}),n)-V(\boldsymbol{\omega},n)\}^{\frac{1}{1-\alpha}}\right\}. \tag{215}$$

退出確率は(199)式，参入確率は(202)式で与えられる。これらを(212)式に代入して $V(\boldsymbol{\omega}, n)$について解くと均衡価値が導かれる。

### 8.2.2　アルゴリズム

すべての企業 $n=1,\cdots,N$, すべての産業構造 $\boldsymbol{\omega}\in S^0$ について，初期値 $V^{(0)}(\boldsymbol{\omega}, n)$, $x^{(0)}(\boldsymbol{\omega}, n)$, $r^{(0)}(\boldsymbol{\omega}, n)$, $\lambda^{(0)}(\boldsymbol{\omega})$を外生的に与えた上で，反復最適反応法により均衡を求める。

$i-1$ 回目の反復で計算された価値関数 $V^{(i-1)}$を使って，$\boldsymbol{\omega}\in S^0$, $n=1,\cdots,N$ について，以下の順番で更新していく。

（1）投資水準の更新

$$x^{(i)}(\boldsymbol{\omega},n)\leftarrow\mathrm{Max}\left\{0,\alpha\{V^{(i-1)}((\omega_n+1,\omega_{-n}),n)-V^{(i-1)}(\boldsymbol{\omega},n)\}^{\frac{1}{1-\alpha}}\right\} \tag{216}$$

（2）退出確率と退出価値の更新

$$r^{(i)}(\boldsymbol{\omega},n)\leftarrow\mathrm{Max}\left\{0,\min\left\{\frac{\phi_H-V^{(i-1)}(\boldsymbol{\omega},n)}{\phi_H-\phi_L},1\right\}\right\} \tag{217}$$

$$\mathrm{E}(\phi\mid\phi\geq V^{(i-1)}(\boldsymbol{\omega},n))\leftarrow\min\left\{\phi_H,\mathrm{Max}\left\{\frac{\phi_H+V^{(i-1)}(\boldsymbol{\omega},n)}{2},\frac{\phi_L+\phi_H}{2}\right\}\right\}^{108)} \tag{218}$$

（3）参入確率と参入価値の更新

$$\lambda^{(i)}(\boldsymbol{\omega})\leftarrow\mathrm{Max}\left\{0,\min\left\{1,\frac{V^{(i-1)}(\boldsymbol{\omega}\cup\omega^e,n)-x_L^e}{x_H^e-x_L^e}\right\}\right\} \tag{219}$$

108) 退出価値は $\phi_H$ より大きくなることはない。

$$V^{e(i)}(\boldsymbol{\omega}) \leftarrow V^{(i-1)}(\boldsymbol{\omega} \cup \omega^e, n) - \mathrm{Max}\left\{ x_L^e, \min\left\{ \frac{V^{(i-1)}(\boldsymbol{\omega} \cup \omega^e, n) + x_L^e}{2}, \frac{x_H^e + x_L^e}{2} \right\} \right\} \quad (220)$$

(220)式右辺第２項は参入費用の期待値 $\mathrm{E}(x^e | x^e \leq V(\omega \cup \omega^e, n))$である[109]。

（４）価値関数の更新

$$\begin{aligned} V^{(i)}(\boldsymbol{\omega}, n) \leftarrow & \frac{1}{\rho + \eta r^{(i)}(\boldsymbol{\omega}, n) + \{x^{(i)}(\boldsymbol{\omega}, n)\}^{\alpha} + \eta^e \lambda^{(i)}(\boldsymbol{\omega}) + \delta + \sum_{m \neq n} [\{x^{(i)}(\boldsymbol{\omega}, m)\}^{\alpha} + \eta r^{(i)}(\boldsymbol{\omega}, m)]} \\ & \times [\pi(\boldsymbol{\omega}, n) - x^{(i)}(\boldsymbol{\omega}, n) + \eta r^{(i)}(\boldsymbol{\omega}, n) \mathrm{E}(\phi | \phi \geq V^{(i-1)}(\boldsymbol{\omega}, n)) \\ & + \{x^{(i)}(\boldsymbol{\omega}, n)\}^{\alpha} V^{(i-1)}((\omega_n + 1, \omega_{-n}), n) + \eta^e \lambda^{(i)}(\boldsymbol{\omega}) V^{(i-1)}(\boldsymbol{\omega} \cup \omega^e, n) \\ & + \sum_{m \neq n} [\{x^{(i)}(\boldsymbol{\omega}, m)\}^{\alpha} V^{(i-1)}((\omega_m + 1, \omega_{-m}), n) + \eta r^{(i)}(\boldsymbol{\omega}, m) V^{(i-1)}((\varnothing, \omega_{-m}), n)] \cdot \end{aligned} \quad (221)$$

この反復式は次のように書くことができる[110]。

$$V^{(i)}(\boldsymbol{\omega}, n) \leftarrow \frac{\pi(\boldsymbol{\omega}, n) - x^{(i)}(\boldsymbol{\omega}, n) + \sum_{\boldsymbol{\omega}'} h_n^{(i-1)}(\boldsymbol{\omega}' | \boldsymbol{\omega}) V^{(i-1)}(\boldsymbol{\omega}', n)}{\rho + \sum_{\boldsymbol{\omega}'} h_n^{(i-1)}(\boldsymbol{\omega}' | \boldsymbol{\omega})} . \quad (222)$$

### 8.2.3　コード[111), 112)]

```
001    while ~done
002      for w = 1:wmax
```

109) 参入費用の期待値は $x_L^e$ よりも小さくなることはない。

110) ベルマン方程式はモジュール $\dfrac{\sum h_n^{(i-1)}(\boldsymbol{\omega}' | \boldsymbol{\omega})}{\rho + \sum h_n^{(i-1)}(\boldsymbol{\omega}' | \boldsymbol{\omega})}$ の縮小写像となり，離散期間モデルのモジュール $\beta$ よりも大きくなる。したがって，連続時間モデルは，1 反復当たりの計算時間は短くなるが，収束するまでに必要とする反復回数は増える。

111) Doraszelski＆Judd アルゴリズムのコードは C 言語で記述されている。

112) ここに掲載したコードは，変数の初期化，許容誤差の設定，計算結果の保存処理などが省略されている。

```
003         di = qdecode(w);
004
005         oldvalue = V1(w, :);
006         oldx = x1(w, :);
007         oldy = y1(w, :);
008
009         for n = 1:nfirms
010           j = 0;
011
012           if (n > 1) && (di(n) == di(n-1))
013             V(n) = V(n-1);
014             x(n) = x(n-1);
015             y(n) = y(n-1);
016             Vspec(n) = Vspec(n-1);
017             rates(n, :) = rates(n-1, :);
018             nz(n, :) = nz(n-1, :);
019
020           elseif di(n) > 0
021             if di(n) < kmax
022               dip = di;
023               dip(n) = di(n)+1;
024               [dip, pos] = sort(dip, 'descend');
025               Vup = V1(qencode(dip), :);
026               diff = Vup(pos(n)) - V(n);
027
028               if diff > 0.0
029                 x(n) = power(alpha*diff, 1.0/(1.0 - alpha));
030               end
031             end
```

```
032
033         temp1 = (Phi_hi - V(n))/(Phi_hi - Phi_lo);
034         y(n) = max(0, min(temp1, 1));
035
036         x(n) = lambda*x(n) + (1.0 - lambda)*oldx(n);
037         y(n) = lambda*y(n) + (1.0 - lambda)*oldy(n);
038
039         temp1 = (Phi_hi + V(n))/2;
040         temp2 = (Phi_hi + Phi_lo)/2;
041         Vspec(n) = min(Phi_hi, max(temp1, temp2));
042
043         if di(n) == 1
044           rates(n, 2) = power(x(n), alpha);
045           j = j+1;
046           nz(n, j) = 2;
047
048         elseif di(n) == kmax
049           rates(n, kmax-1) = delta;
050           j = j+1;
051           nz(n, j) = kmax-1;
052
053         else
054           rates(n, di(n)-1) = delta;
055           rates(n, di(n)+1) = power(x(n), alpha);
056           j = j+1;
057           nz(n, j) = di(n)-1;
058           j = j+1;
059           nz(n, j) = di(n)+1;
060         end
```

```
061
062         if y(n) > 0.0
063           rates(n, kmax+1) = etax*y(n);
064           j = j+1;
065           nz(n, j) = 0;
066         end
067
068         j = j+1;
069         nz(n, j) = kmax+1;
070
071       else
072         x(n) = 0.0;
073
074         dip = di;
075         dip(n) = we0;
076         [dip, pos] = sort(dip, 'descend');
077
078         Vin = V1(qencode(dip), :);
079         temp1 = (Vin(pos(n)) - x_entryl)/(x_entryh - x_entryl);
080         y(n) = max(0, min(1, temp1));
081
082         x(n) = lambda*x(n) + (1.0-lambda)*oldx(n);
083         y(n) = lambda*y(n) + (1.0-lambda)*oldy(n);
084
085         temp1 = (Vin(pos(n)) + x_entryl)/2;
086         temp2 = (x_entryh + x_entryl)/2;
087         Vspec(n) = Vin(pos(n)) - max(x_entryl, min(temp1, temp2));
088
089         if y(n) > 0.0
```

```
090             rates(n, entry_k) = etae*y(n);
091             j = j+1;
092             nz(n, j)= entry_k;
093           end
094
095           j = j+1;
096           nz(n, j) = kmax+1;
097         end
098       end
099
100       SV = zeros(1, nfirms);
101       SH = zeros(1, nfirms);
102
103       for n = 1:nfirms
104         for m = 1:4
105           in = nz(n, m);
106           if in < kmax+1
107             dip = di;
108             dip(n) = in;
109             [dip, pos] = sort(dip, 'descend');
110             V = V1(qencode(dip), :);
111
112             for k = 1:nfirms
113               if di(k) > 0
114                 if (n == k) && (in == 0)
115                   SV(k) = SV(k) + Vspec(k)*rates(n, kmax+1);
116
117                 elseif (n ~= k) && (in == 0)
118                   SV(k) = SV(k) + V(pos(k))*rates(n, kmax+1);
```

```
119
120                 else
121                   SV(k) = SV(k) + V(pos(k))*rates(n, in);
122                 end
123
124                 if in == 0
125                   SH(k) = SH(k) + rates(n, kmax+1);
126                 else
127                   SH(k) = SH(k) + rates(n, in);
128                 end
129               end
130             end
131
132           else
133             break;
134           end
135         end
136       end
137
138       for n = nfirms:-1:1
139         if di(n) == 0
140           SV(n) = SV(n) + Vspec(n)*rates(n, entry_k);
141           SH(n) = SH(n) + rates(n, entry_k);
142         else
143           break;
144         end
145       end
146
147       for n = 1:nfirms
```

```
148          if di(n) > 0
149            V(n) = (profit(w, n) - x(n) + SV(n))/(rho + SH(n));
150          else
151            V(n) = SV(n);
152          end
153
154          V(n) = lambda*V(n) + (1.0 - lambda)*oldvalue(n);
155        end
156
157        %誤差を計算する%
158
159       V1(w, :) = V;
160       x1(w, :) = x;
161       y1(w, :) = y;
162     end
163
164     if %誤差が許容誤差より小さくなる%
165       done = 1;
166     end
167
168     if ~done
169       iter = iter+1;
170     end
171   end
```

反復を繰り返すことで，均衡投資水準と均衡価値，および均衡における参入確率・退出確率の推測値を改良し続け，マルコフ完全ナッシュ均衡値へと収束させる。１回の反復ですべての産業構造を巡回し，それぞれの産業構造に対してすべての企業を巡回して新しい推測値を計算する。前回の反復で導かれた古い推測値

と比較して，両者の差が許容誤差よりも小さくなるとき，推測値が均衡値に収束したと判断して反復を終了する。

計算された均衡価値，均衡投資水準，および均衡における参入確率・退出確率の推測値は，ルックアップ・テーブル V1，x1，y1 にそれぞれ保存される。反復処理において，これらの値を必要とするときは，ルックアップ・テーブルを参照することで値を取り出す。これらのルックアップ・テーブルはすべて wmax×nfirms 行列である。同一企業で参入と退出が同時に起こることはないので，参入確率と退出確率は同じルックアップ・テーブル y1 に格納する。

001行目：done が 0 である限り推測値は収束していないので，反復処理を続ける。
002行目：産業構造 w についてループする。
003行目：産業構造 w をデコードして産業構造ベクトル di を得る。
005～007行目：前回の反復で導かれた産業構造 w における均衡投資水準，均衡価値，および均衡における参入確率・退出確率の推測値をルックアップ・テーブルから参照して，それぞれ oldvalue，oldx，oldy に代入する。

009～098行目は企業 n に関するループである。020～069行目は既存企業に関する処理であり，投資水準，退出確率，退出価値，およびハザード率を計算する。
010行目：カウンタ j は，正の値を持つハザード率の個数を数え上げる。このカウンタを 0 に戻す。
012～018行目：前回の反復で得た結果について，対称性の制約（脚注 12) 参照）を課す。効率性 di(n－1)と同じ水準の効率性を持つ企業 n について，価値関数 V，投資関数 x，参入確率・退出確率 y，参入価値・退出価値 Vspec，およびハザード率の情報を格納した行列 rates, nz を，企業 n－1のそれらと同じ水準にする。
020行目：0 よりも大きい効率性を持つ企業は既存企業である。
021行目：最大水準の効率性 kmax を持たない企業は，投資により効率性が上昇する可能性があるので，これらの企業の投資水準を計算する。
022行目：dip は企業 n の効率性が上昇したときの産業構造ベクトルである。まず，di を dip にコピーする。

023行目：上昇した企業 n の効率性 di(n) + 1を dip の第 n 要素に代入する。
024行目：dip を降順に並び替えたものを dip に上書きする。降順に並び替えた際に企業のインデックスも入れ替わるが，この並び替えられた企業のインデックス・ベクトルを pos とする。企業 n のインデックスは pos(n)となる。
025行目：qencode(dip)は dip をエンコードしたものである。価値関数のルックアップ・テーブルの第 qencode(dip)行を参照して，産業構造 dip における各企業の価値を Vup に代入する。
026行目：効率性が上昇したときの価値の変化 diff を計算する。
028～030行目：価値の変化 diff が正であるとき，(216)式に従って均衡投資水準を計算する。
033，034行目：(217)式に従って退出確率を計算する。
036，037行目：収束しやすくするために，投資水準と退出確率を減衰（dampen）する。つまり，古い推測値と新しい推測値を凸結合した値を，改めて新しい均衡の推測値とする。反復によって推測値が大きく変化すると収束が遅くなるので，凸結合により推測値の変化を円滑化することで収束しやすくする。
039～041行目：(218)式に従って退出価値 E($\phi|\ \phi > V(\omega, n)$)を計算する。

043～069行目で既存企業のハザード率を計算する。ジャンプした先の効率性を nz 行列に，このジャンプが発生するハザード率を rates 行列にそれぞれ格納する。企業 n に関するこれらの値は，２つの行列の第 n 行に格納する。効率性$\omega$へジャンプするハザード率が正であるならば，そのハザード率を rates 行列の第$\omega$列に格納する。このハザード率を格納したらカウンタ j を１つ増やし，効率性$\omega$を nz 行列の第 j 列に格納する。別のジャンプを考えて，そのハザード率が正であるならばカウンタを１つ増やして同様の作業を行う。このようにして，カウンタ j を正のハザード率を持つジャンプの１つに対応させる。
043～046行目：最低水準の効率性１を持つ企業 n について，効率性はこれより低い水準に下落することはなく，効率性２に上昇するだけである。したがって，効率性２へのジャンプが発生するハザード率は正であるため，効率性上昇のハザード率(205)式を rates 行列の第２列に格納する。カウンタ j を１つ増やして，nz

行列の第 j 列に効率性２を格納する。
048～051行目：最高水準の効率性 kmax を持つ企業 n について，効率性はこれより高い水準に上昇せず，効率性 kmax－1に下落するだけである。したがって，効率性 kmax－1へのジャンプが発生するハザード率は正であるため，効率性下落のハザード率$\delta$を rates 行列の第 kmax－1列に格納する。カウンタ j を１つ増やして，nz 行列の第 j 列に効率性 kmax－1を格納する。
053～060行目：企業 n の効率性が１よりも大きく kmax 未満であるとき，効率性の上昇・下落ともに起こりうるので，それらのハザード率は正となり，rates 行列の第 di(n) + 1列および第 di(n) － 1列にそれぞれハザード率を格納する。カウンタ j を１つずつ増やして，nz 行列の第 j 列に効率性 di(n) + 1および効率性 di(n)－1をそれぞれ格納する。
062～066行目：退出確率が正であるとき退出発生のハザード率も正となるから，rates 行列にハザード率(203)式を格納する。退出した企業の効率性は空きスロット０であるが，MATLAB®がそうであるように，０のインデックスが存在しないプログラミング言語の場合は，rates 行列の kmax + 1列にハザード率を格納することで対処する。カウンタ j を１つ増やして，nz 行列の第 j 列に効率性０を格納する。
068，069行目：既存企業 n について，正の値を持つハザード率はすべて rates 行列に格納したので，カウンタを１つ増やして，nz 行列の第 j 列に番兵 kmax＋1を配置する。nz 行列の第 j＋1列以降の列の要素はすべて kmax＋1である[113]。

071～097行目は参入企業に関する処理であり，投資水準，参入確率，参入価値，およびハザード率を計算する。
072行目：参入直後の投資水準はゼロである。
074～076行目：参入企業 n の効率性は entry_k である。dip の第 n 要素に entry_k を代入した上で降順に並べ替えることで，参入発生後の産業構造 dip を得る。また，並べ替えられた企業のインデックス・ベクトル pos を得る。参入

113) nz 行列の初期化で要素をすべて kmax＋1 にしておく。

企業のインデックスは pos(n)となる。
078行目：産業構造 dip をエンコードして qencode(dip)を得る。そして，ルックアップ・テーブルの第 qencode(dip)行を参照して，参入後に獲得する期待割引現在価値を Vin に代入する。
079，080行目：(219)式に従って参入確率を計算する。
082，083行目：参入企業の投資水準と参入確率を減衰して新しい推測値とする。
085～087行目：(220)式に従って参入価値を計算する。
089～093行目：参入確率が正であるならば参入発生のハザード率も正となるから，rates 行列の第 entry_k 列にハザード率(204)式を格納し，カウンタ j を 1 つ増やしたうえで，nz 行列の第 j 列に効率性 entry_k を格納する。
095，096行目：参入企業 n について，正の値を持つハザード率はすべて rates 行列に格納したので，カウンタを 1 つ増やして，nz 行列の第 j 列に番兵 kmax+1 を配置する。

既存企業および参入企業の均衡価値を計算する。効率性がジャンプする企業 n に関するループを考えて，その企業のカウンタ m を 1 つずつ増やす。それぞれのカウンタに対応する正のハザード率を rates 行列から，ジャンプ先の効率性を nz 行列から順次取り出すと，ジャンプ発生後の産業構造$\boldsymbol{\omega}'$とそのハザード率が判明する。これらの値を用いて企業 k について SV = $\sum h_k(\boldsymbol{\omega}'|\,\boldsymbol{\omega})V(\boldsymbol{\omega}', n)$と SH = $\sum h_k(\boldsymbol{\omega}'|\,\boldsymbol{\omega})$を計算して，それから(222)式に従って企業 k の均衡価値を計算する。
100，101行目：SV と SH を初期化する。
103，104行目：効率性がジャンプする企業 n について，m 番目のジャンプを考える。
105行目：nz 行列の第 n 行，第 m 列を in とする。m 番目のジャンプは企業 n の効率性が in にジャンプするものである。
106行目：in が番兵 kmax＋1未満の値であるならば，効率性 in へのジャンプが発生するハザード率は正となる。
107～109行目：dip の第 n 要素に効率性 in を代入して，降順に並び替えたものをジャンプ発生後の産業構造として，改めて dip とする。並び替えられた企業の

インデックス・ベクトルを pos とする。自社のインデックスは pos(k)となる。
110行目：dip をエンコードすると qencode(dip)となる。ルックアップ・テーブルの第 qencode(dip)行を参照して，ジャンプ発生後の各企業の継続価値を V に格納する。
112行目：自社のインデックスは k である。k についてループする。
113～115行目：自社は既存企業であり，自社が退出するときの SV を計算する。退出価値は Vspec であり，退出発生のハザード率は rates 行列の第 kmax + 1 列に格納されている。
117, 118行目：他社が退出するときの SV を計算する。退出後の産業構造は dip に変わり，自社のインデックスは pos(k)となるので，継続価値は V(pos(k))となる。
120, 121行目：どの企業も退出しないときの SV を計算する。発生するジャンプは，企業 n の効率性が in に上昇するか，産業全体ショックにより企業 n の効率性が in に低下するか，あるいは企業 n の参入により空きスロットに効率性 in が埋まるか，のいずれかである。
124～128行目：SH を計算する。rates 行列からハザード率を取り出して加えていく。自社が退出するときのハザード率は第 kmax + 1列に格納されている（124, 125行目）。それ以外のハザード率は，ジャンプする効率性 in と同じ数字の列番号に格納されている（126, 127行目）。
132, 133行目：in が番兵 kmax + 1と同じ値になるときは，正のハザード率を持つジャンプはすべて検討したことを意味するので，カウンタ m のループから抜け出す。
138, 139行目：参入企業について SV と SH を計算する。産業構造ベクトルは効率性が降順に並べられているので，nfirms から順にインデックス n を減らしていき，企業 n の効率性 di(n)が 0 であるかどうか調べる。0 ならば企業 n は参入企業である。
140, 141行目：参入企業に発生するジャンプは，空きスロットに効率性 entry_k が埋まるジャンプだけである。したがって，rates 行列の第 n 行，第 entry_k 列からハザード率 $\eta' \lambda(\omega)$ を取り出して SH に加える（141行目）。また，SV に

$V^e(\omega)\times\pi^e\lambda(\omega)$を加える（140行目）。
142，143 行目：企業 n の効率性が 0 でないとき，それより小さいインデックスを持つ企業はすべて既存企業となるので，n のループから抜け出す。
147～155 行目：(222) 式に従って，既存企業の均衡価値を計算する（149 行目）。また参入企業の均衡価値は参入前の期待価値とするので，参入発生のハザード率も考慮に入れて $V^e(\omega)\times\pi^e\lambda(\omega)$とする（153 行目）。そして，均衡価値を減衰する（155 行目）。

159～161行目：こうして計算された産業構造 w における均衡価値，均衡投資水準，均衡における参入確率・退出確率を，ルックアップ・テーブル V1，x1，y1の第 w 行にそれぞれ格納する。
164～166行目：収束判定をする。収束したと判断されたときは，done に 1 を代入する。
168～170行目：done が 0 であるときは収束していないので，反復回数を 1 つ増やして，再び反復を行う。

### 8.2.4　ガウス＝ザイデル法

マルコフ完全ナッシュ均衡を求めることは，均衡条件(77)，(78)，(79)，(96)，(100) 式で定義される $V(\omega, n)$，$x(\omega, n)$，$V^e(\omega)$，$\lambda(\omega)$（$\omega\in\Omega$，$n=1,\cdots,N$）の非線形連立方程式を解くことと同値である[114]。Pakes&McGuire アルゴリズムは，ガウス＝ヤコビ法により非線形連立方程式を解いている。ガウス＝ヤコビ法では，$i-1$ 回目の反復で推測値 $A^{(i-1)}(\omega, n)$（$\omega\in\Omega, n=1,\cdots,N$）をすべて計算した後に，$i$ 回目の反復で，これらの値を使用して新しい推測値 $A^{(i)}(\omega, n)$（$\omega\in\Omega, n=1,\cdots,N$）を計算する。これに対して，Doraszelski&Judd アルゴリズムは，ガウス＝ザイデル法で非線形連立方程式を解いている。encode($\omega^\dagger$) < encode($\omega$)のとき，$\omega^\dagger$の推測値は$\omega$の推測値よりも先に計算される。ガウス=ザイデル法では，$i$ 回目の反復で$\omega$の推測値を計算するときに，$\omega^\dagger$の推測値が必要になるならば，最新の推測値

114) 方程式の数が膨大であるので一気に解くことができない。そこで，連立方程式を小部分に分割してから，分割された連立方程式を別々に解く。

$A^{(i)}(\boldsymbol{\omega}^{\dagger}, n)$を使用する[115]。

C 言語ではポインタ変数が利用できるので，ガウス＝ザイデル法が自然な形で実行される。V，x，y は配列の先頭アドレスを指すポインタ，また，V1，x1，y1 はルックアップ・テーブルの先頭アドレスを指すポインタになる。V ← V1，x ← x1，y ← y1 とポインタをコピーすることで，V，x，y と V1，x1，y1 はアドレスを共有し，したがって同じ値を参照するようになる。こうすることで，V，x，y の値が更新されると，ルックアップ・テーブルの内容も同時に更新される[116]。したがって，$\boldsymbol{\omega}$の推測値 $A^{(i)}(\boldsymbol{\omega}, n)$を計算するときに，ルックアップ・テーブルから$\boldsymbol{\omega}^{\dagger}$の値を参照すると，新しい推測値 $A^{(i)}(\boldsymbol{\omega}^{\dagger}, n)$が取り出される。

## 9．ま と め

一連の研究ノートでは，マルコフ完全産業動学の理論モデル（Ericson&Pakes モデル），および数値計算モデル（Pakes&McGuire アルゴリズム）について解説してきた。Ericson&Pakes モデルは拡張可能性が高く，一般性を備えた産業動学モデルとして注目されてきた。しかし，均衡を陽表的に求めることはほぼ不可能であるので，需要関数や費用関数，また確率分布などを特定化し，パラメータに具体的な値を割り当てた上で，コンピュータを援用した数値計算により均衡を求めるという方法が取られる。数値計算ができるように Ericson&Pakes モデルをアレンジして，それを基にアルゴリズムを構築したものが Pakes&McGuire アルゴリズムである。この画期的なアルゴリズムに様々な改良が加えられ，計算の高速化や効率化が図られた。Pakes&McGuire アルゴリズム（＝ソフトウェア）が進化すると共に，CPU（＝ハードウェア）が日進月歩で高性能になり，数値計算を牽引する両輪の技術が大きく発展した[117]。その結果，計算速度が飛躍的に上昇し，計

115) 一般的に，ガウス＝ヤコビ法よりもガウス＝ザイデル法の方が収束するスピードが速い。
116) 159～161 行目のコードは不要になる。
117) 今後の展望として，ソフト面では，遺伝アルゴリズムや機械学習アルゴリズムを産業動学モデルへ応用することで，解探索の効率化と高速化が期待される。また，ハード面では，量子コンピュータが汎用化・実用化することで，計算速度の超高速化と計算コストの超低廉化が期待される。

算コストが大幅に低下した。こうして現在では，規模の大きな最適化問題である産業動学モデルについて，パーソナル・コンピュータでも比較的短時間で均衡が計算できるようになった。このような流れもあり，産業組織論の理論研究においても，数値計算による分析が益々盛んになるものと予想される[118]。

最後に，数値計算による理論研究の意義について言及して一連の研究ノートを閉じる。産業組織論における理論研究の多くは，演繹分析（deductive analysis）により議論を展開して結論に至るという方針が取られている。すなわち，構築されたモデルを踏まえて，定理の証明という形で論証したり，代数演算や微積分などの演算を繰り返したりすることで，経済学的な知見を導き出す。特に後者の手法を取るときは，一般モデルを解析学的に扱いやすい（analytically tractable）形に特定化した上で分析することが多い。例えば，産業組織論で扱われる動学モデルは，その多くが２期間でモデルを構築し，費用関数と需要関数を線形・二乗（linear-quadratic）型で特定化して分析している。モデルをこのように特定化する理由は，解を陽表的に求めることができるからである。モデルは他にも様々な形で特定化できるにも関わらず，解析学的に扱いやすいという一側面だけで，線形・二乗型２期間モデルという特殊なサンプルを採用して分析しているのである。言い換えれば，線形・二乗型２期間モデルは，一般モデルを特定化した様々なサンプルの中の１つに過ぎず，またそれが一般モデルの本質を完全に表現するような代表的サンプルであるわけでもない。このことから，線形・二乗型２期間モデルを演繹的に分析することで導かれた結論が，一般モデルを特徴付ける包括的な結論であると主張することはできない。

数値計算分析（computational analysis）は，計算可能なケースに制限されるものの，２期間以上の有限期間モデルや無限期間モデルで分析することができるし，様々な型の需要関数や費用関数や確率分布，あるいは様々な形態の企業間競争でモデルを設定して分析することもできる[119]。さらに，パラメータを連続的に変化

118) 数値計算を用いた産業組織論の先駆的な理論研究として，Quirmbash (1993) が挙げられる。

119) Pakes&McGuire アルゴリズムでは，静学・動学分解（脚注 62) 参照）を前提としているので，ベルトラン競争やクールノー競争といった静学ゲームの設定は，オフラインで処理される。

させることで，同じ型のモデルについて幅広いサンプルを抽出することができる。極限すれば，数値計算分析の目的は，一般モデルを特定化した多種多様で膨大なサンプルについて均衡値を求めて，それらがたどる共通のパターンを見出すことに尽きる。数値計算分析は演繹分析と比べて膨大なサンプルを抽出して分析している点で，一般モデルが備えている特徴を反映した，より一般的で頑健な結論を導き出す分析手法になる。

演繹分析で導かれる結果は定性的なものが殆どであるが，数値計算分析の結果は定量的であり，均衡が具体的な数値として出力される。このことから，定量分析は定性分析で明らかにすることができない新たな知見を生み出す。例えば，均衡が複雑なパターンを示すために，その特徴を簡潔かつ厳密な定理として記述できないことがある。定性分析は，本来複雑である均衡を単純化して記述する傾向にあり，一般モデルが持つ本質的な特徴を捨象することにもなりかねない。これに対して定量分析では，均衡が示す複雑な振る舞いを数値の変化として捉えることができる。また，定量的な結果は，現実の統計データと突き合わせることで，その妥当性を検討することもできる。

（終）

**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] 奥村晴彦（1991）『C 言語による最新アルゴリズム事典』，技術評論社。
[3] 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。
[4] 蓑谷千凰彦（1998）『すぐに役立つ統計分布』，東京図書。
[5] 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.
[6] Doraszelski, U. and K. Judd. (2012), “Avoiding the Curse of Dimensionality in Dynamic Stochastic Games.”, *Quantitative Economics*, Vol.3, pp.53-93.
[7] Ericson, R. and A. Pakes. (1995), “Markov-Perfect Industry Dynamics: A Framework for Empirical Work.”, *Review of Economic Studies*, Vol.62, pp.53-82.
[8] Goettler, R. and B. Gordon. (2014), “Competition and Product Innovation in Dynamic Oligopoly.”, *Quantitative Marketing Economics*, Vol.12, pp.1-42.
[9] Judd, K. (1998), *Numerical Methods in Economics*, MIT Press.
[10] Mathews, J. and K. Fink. (2003), *Numerical Methods Using MATLAB*, Pearson Prentice-Hall.

[11] Miranda, M. and P. Fackler. (2002), *Applied Computational Economics and Finance*, MIT Press.
[12] 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.
[13] Press, W., S. Teukolsky., W. Vetterling., and B.Flannery. (2007), *Numerical Recipes*: *The Art of Scienctific Computing*, Cambridge University Press.
[14] Quirmbash, H. (1993), “R&D: Competition, Risk, and Performance.”, *RAND Journal of Economics*, Vol.24, pp.157-197.
[15] Stachurski, J. (2009), *Economic Dynamics*: *Theory and Computation*, MIT Press.
[16] Stokey, N. and R. Lucas, Jr. (1989), *Recursive Methods in Economic Dynamics*, Harvard University Press.