『MPSGEにおける技術進歩の導入方法』

武田史郎

関東学園大学経済学部
373–8515 群馬県太田市藤阿久町 200

2009/12/01

概 要
生産性を表すパラメータを導入したモデルを MPSGE のコードで記述する方法。

1 準備

以下の議論は CES 関数のケースにも適用できるが、ここでは例として Cobb-Douglas 型生産関数を利用して議論を進める。

$$q = \phi \prod_i [\lambda_i y_i]^{\alpha_i} \tag{1}$$

q は生産量、y_i は投入物 i の投入量、ϕ、α_i ($\sum_i \alpha_i = 1$) は定数である。また、λ_i は投入物 i に関する生産性パラメーターであり、値が大きくなるほど生産性が高いことになる。この λ_i の初期値は 1 とする。

生産関数が (1) 式で与えられる場合、単位費用関数、単位需要関数はそれぞれ以下のような形式となる。

$$c = \frac{1}{\phi} \prod_i \left[\frac{p_i}{\lambda_i \alpha_i} \right]^{\alpha_i} \tag{2}$$

$$x_i = \frac{\alpha_i}{p_i} c \tag{3}$$

p_i は投入物 i の価格である。

さらに、以上の関数を、ϕ、α_i を消した calibrated share form で書き直すと、

$$q = \bar{q} \prod_i \left[\frac{x_i}{\bar{x}_i / \lambda_i} \right]^{\theta_i} \tag{4}$$

$$c = \bar{c} \prod_i \left[\frac{p_i}{\lambda_i \bar{p}_i} \right]^{\theta_i} \tag{5}$$

$$x_i = \frac{\theta_i}{p_i} c \tag{6}$$

となる[1]。"-" 付きの変数はその変数のベンチマーク値を表す。また、θ_i は benchmark における投入シェアであり

$$\theta_i \equiv \frac{\bar{p}_i \bar{x}_i}{\sum_j \bar{p}_j \bar{x}_j} \tag{7}$$

[1] (3) 式、及びベンチマークの $\bar{p}_i$、$\bar{x}_i$、$\bar{c}$ より α_i がカリブレートできる。

$$\alpha_i = \frac{\bar{p}_i \bar{x}_i}{\bar{c}} = \frac{\bar{p}_i \bar{x}_i}{\sum_j \bar{p}_j \bar{x}_j} = \theta_i$$

である。

2 MPSGE のコード

以上の準備をもとに (1) の MPSGE での表現を考える。これまでの記号を MPSGE コードでは次のように表現する。

q	→	`q`
c	→	`c`
p_i	→	`p(i)`
λ_i	→	`lambda(i)`
$\bar{q}$	→	`q0`
$\bar{x}_i$	→	`x0(i)`
$\bar{p}_i$	→	`p0(i)`

まず、技術進歩を全く考慮しないケースでは

```
$prod:q         s:1
    o:c         q:q0
    i:p(i)      q:x0(i)         p:p0(i)
```

のような形式になる。ここに生産性パラメータ `lambda(i)` を導入するのであるが、(4) 式を見ると生産性パラメータはベンチマークの生産量を割る形で入ってきているので、以下のように書き換えればよいのではないかと考えられる[2]。

[Case 1]

```
$prod:q         s:1
    o:c         q:q0
    i:p(i)      q:(x0(i)/lambda(i))         p:p0(i)
```

この記述は一見もっともらしく感じるが、実はこれでは技術進歩を正確に導入したことにはならない。その理由は MPSGE でのパラメータのカリブレーション方法にある。MPSGE では投入物の reference quantity と price を利用してベンチマークにおける投入シェアを計算する。そして、(1) 式の α_i はその投入シェアによってカリブレートされることになる。

ここで、上のように MPSGE コードを記述したとき、投入シェアがどう計算されるか求めてみると

$$\alpha_i = \theta_i = \frac{\bar{p}_i \bar{x}_i / \lambda_i}{\sum_j \bar{p}_j \bar{x}_j / \lambda_j} \tag{8}$$

また、ϕ は

$$\phi = \bar{q} / \prod_i (\bar{x}_i)^{\theta_i}$$

でカリブレートできる。

[2] `lambda(i)` は値が大きいほど生産性が高くなるように定義されているので、ベンチマークの生産量を割る形式で導入される。

となる。この式から明かなように、この場合ベンチマークの投入シェアが λ_i の値に依存することになる[3]。これは生産性パラメータを変化させることで (1) 式の α_i をも変化させてしまうことを意味する。これでは純粋に生産性の変化だけを導入したことにはならない。

以上のように、MPSGE のカリブレーション方法では reference quantity 側だけに生産性パラメータを導入してしまうと、生産性の変化だけではなく、シェアパラメータの α_i までも変化させることになるという問題が生じる。

それでは、純粋に生産性の変化だけを導入する際にはどのようにコードを記述すべきかというと、そのヒントは calibrated share form の単位費用関数 (5) 式に含まれている。(5) では reference price にあたる部分が $\bar{p}_i\lambda_i$ に置き換えられている。これと同じように、MPSGE のコードでも reference price の部分を書き換えればよいのである。すなわち

[Case 2]

```
$prod:q         s:1
    o:c         q:q0
    i:p(i)      q:(x0(i)/lambda(i))        p:(p0(i)*lambda(i))
```

とすればよい。

このようにすれば reference quantity、price から計算される投入シェアは

$$\alpha_i = \theta_i = \frac{(\bar{p}_i\lambda_i)(\bar{x}_i/\lambda_i)}{\sum_j(\bar{p}_j\lambda_j)(\bar{x}_j/\lambda_j)} = \frac{\bar{p}_i\bar{x}_i}{\sum_j \bar{p}_j\bar{x}_j} \tag{9}$$

となり、λ_i には依存しなくなるため、α_i に影響を与えることなく、生産性パラメータの変化の効果のみをとりだすことができる。

3 MPSGE の sample code

```
$title  A sample MPSGE program including technological change:
$ontext
Time-stamp:     <2009-12-01 21:28:14 Shiro Takeda>
First-written:  <2006/07/17>

[Two experiments]
Case 1 -> Reference quantity の生産性パラメータを上昇
Case 2 -> Reference quantity & price の生産性パラメータを上昇

Case 2 が適切に技術進歩を考慮したケース。

[Results]
Case 1 と case 2 で生産量の伸び率が異なってくる。
Case 1 の場合、Cobb-Doublas 型関数であるのに、要素の投入シェアが変わってしまう。
```

[3] λ_i が全て等しいという特殊ケースは除く。

```
$offtext

*       Set definition:
set     f       Index of factor / k, l /;
display f;

*       Parameter definition:
parameter
    q0          Reference output
    x0(f)       Reference input
    p0          Reference price of output
    pf0(f)      Reference price of factors
    end0(f)     Endowment of factors

    lambda_q(f) Efficiency parameter
    lambda_p(f) Efficiency parameter
;
q0 = 100;
x0("k") = 0.3 * q0;
x0("l") = q0 - x0("k");
p0 = 1;
pf0(f) = 1;
end0(f) = x0(f);
lambda_q(f) = 1;
lambda_p(f) = 1;
display q0, x0, p0, pf0, end0, lambda_q, lambda_p;

*       ---------------------------------------------------------------------
*       MPSGE code:
$ontext
$model:sample

$sectors:
    q           ! Output

$commodities:
    p           ! Output price
    pf(f)       ! Price of primary factor

$consumers:
    ra          ! Representative agent

$prod:q         s:1
```

```
    o:p         q:q0
    i:pf(f)     q:(x0(f)/lambda_q(f))   p:(pf0(f)*lambda_p(f))

$report:
    v:x(f)      i:pf(f)         prod:q  ! Quantity of input

$demand:ra
    d:p         q:q0
    e:pf(f)     q:end0(f)

$offtext

$sysinclude mpsgeset sample

parameter
    result      Compare two results;

*       ---------------------------------------------------------------------
*       Benchmark replication.
sample.iterlim = 0;
$include sample.gen
solve sample using mcp;
result("Bench","q") = q.l;
result("Bench","Sh_k") = pf.l("k") * x.l("k") / (sum(f, pf.l(f) * x.l(f)));
result("Bench","Sh_l") = pf.l("l") * x.l("l") / (sum(f, pf.l(f) * x.l(f)));
result("Bench","v_k") = pf.l("k") * x.l("k") / p.l;
result("Bench","v_l") = pf.l("l") * x.l("l") / p.l;
sample.iterlim = 9000;

*       ---------------------------------------------------------------------
*       Increase only in lambda_q:
lambda_q("k") = 2;

$include sample.gen
solve sample using mcp;
result("Case1","q") = q.l;
result("Case1","Sh_k") = pf.l("k") * x.l("k") / (sum(f, pf.l(f) * x.l(f)));
result("Case1","Sh_l") = pf.l("l") * x.l("l") / (sum(f, pf.l(f) * x.l(f)));
result("Case1","v_k") = pf.l("k") * x.l("k") / p.l;
result("Case1","v_l") = pf.l("l") * x.l("l") / p.l;

*       ---------------------------------------------------------------------
*       Increase both in lambda_q and lambda_p:
```

```
lambda_q("k") = 2;
lambda_p("k") = 2;

sample.iterlim = 9000;
$include sample.gen
solve sample using mcp;
result("Case2","q") = q.l;
result("Case2","Sh_k") = pf.l("k") * x.l("k") / (sum(f, pf.l(f) * x.l(f)));
result("Case2","Sh_l") = pf.l("l") * x.l("l") / (sum(f, pf.l(f) * x.l(f)));
result("Case2","v_k") = pf.l("k") * x.l("k") / p.l;
result("Case2","v_l") = pf.l("l") * x.l("l") / p.l;

display result;

* --------------------
* Local Variables:
* mode: gams
* fill-column: 90
* End:
```

---- 448 PARAMETER result Compare two results

	q	Sh_k	Sh_l	v_k	v_l
Bench	1.000	0.300	0.700	30.000	70.000
Case1	1.130	0.176	0.824	19.943	93.068
Case2	1.231	0.300	0.700	36.934	86.180