GRIPS Discussion Paper 10-34

契約理論：プログラミング・モデル・アプローチ

Contract Theory: A Programming- Model Approach

橋本日出男，濱田弘潤，細江宣裕
Hideo Hashimoto, Kojun Hamada, Nobuhiro Hosoe

2011 年 2 月

GRIPS
NATIONAL GRADUATE INSTITUTE FOR POLICY STUDIES

National Graduate Institute for Policy Studies
7-22-1 Roppongi, Minato-ku,
Tokyo, Japan 106-8677

契約理論:プログラミング・モデル・アプローチ

2011 年 5 月 11 日

橋本日出男[a], 濱田弘潤[b], 細江宣裕[c]

要約:

本稿は、伊藤(2003, 第 1 章)の「部品調達問題」についての数値計算モデルを構築し、契約理論モデルの特性について数値例によって理解を深めようとするものである。前半では伊藤に従って 2 タイプと 3 タイプのケースを考察し、後半では理論モデルに導入された簡単化のための仮定を外した場合や、簡単化の仮定が成立する状況がどの程度起こりそうであるかについても検討する。さらに、数値計算モデルの優位性を生かして、タイプ数が非常に多い場合であってもモデルを容易に拡張・適用できることを示す。

キーワード:

プリンシパル=エージェント問題, 逆選択, 数値計算モデル, 単一交差性, 単調性

---

[a] 大阪大学名誉教授, 政策研究大学院大学客員研究員

[b] 新潟大学経済学部准教授

[c] 政策研究大学院大学准教授. 連絡先: 106-8677 東京都港区六本木 7-22-1 政策研究大学院大学, E-mail: nhosoe@grips.ac.jp

1.　プログラミング・モデル・アプローチについて

本稿は、伊藤(2003)の第 1 章「アドバース・セレクション」が示すプリンシパル=エージェント問題(部品調達問題)を数値的に解く手法について論じる。同書は、契約理論を一般的かつ厳密に展開しているために、初学者にとって理解しにくいところがある。そのため、そこで示されている問題を数値問題として作って解いていくことによって、契約理論をよりよく理解しようとするのが、本稿の目的である。

具体的には、メーカー(プリンシパル)が、複数タイプ存在するサプライヤー(エージェント)と部品供給契約を結ぶ問題を考える。伊藤の議論に従って、順次、サプライヤーのタイプが 2 個のモデルと 3 個のモデルに対応したコンピュータ・プログラムを作っていく。それぞれの場合について、情報の非対称性のないファーストベスト・モデルと、それがあるセカンドベスト・モデルのプログラムを作る。セカンドベスト・モデルは、サプライヤーが自分以外のサプライヤーであると偽る可能性があり、これを防ぐための誘因両立制約が必要となる。タイプ数が増えるにつれて、誘因両立制約が複雑になる。この複雑性を避けるために、しばしば理論分析ではいくつかの仮定を設けて理論モデルの簡単化を図る。本稿では、こうした簡単化の仮定を課した場合はもとより、これらが成立しない場合についても数値例を用いて考察する。さらに、数値計算モデルの優位性を生かし、タイプ数が非常に多くなった(たとえば、10 個)場合についても検討する。

数値モデルを解くために、数値計算ソフトウェアとして、GAMS(General Algebraic Modeling System)を用いることにし、そのためのプログラムを提示する。[1]GAMS の文法に関しては、本稿中のプログラムに即して最低限の説明を施しているが、より詳細な文法については細江他(2004, 第 3 章)が簡便で有用であろう。各自が GAMS 試用版を使って、本稿で示すプログラムをそのまま、あるいは、自由にパラメータなどを変更して、契約理論の要諦について数値例を用いて「実感」することができる。

---

[1] GAMS は市販のソフトウェアであるが、その試用版は、解くことができるモデルの大きさに制限があるものの、GAMS Development Corporation の Web サイト(http://www.gams.com/dowmload/)からダウンロードして無料で利用できる。なお、本稿で取り扱う問題は、この試用版で解くことができる程度に小さい問題であるので、この試用版で十分である。

もちろん、種々の特定化を施された関数型や、仮定されたパラメータ設定の下で作られた数値例である以上、その結果から一般的な結論を導くことには慎重にならざるを得ない。技術的には、数値計算モデルを構築するために新しいコンピュータ言語を習得する手間も小さくはない。その一方で、理論モデルのエッセンスを数値例を用いて表現するわれわれの手法—プログラミング・モデル・アプローチと呼ぼう—によって、わかりやすさという恩恵を得ることができる。また、一般的な設定の下で理論的検討を行うために導入されたいくつかの技術的な仮定から解放されるという利点もある。われわれとしては、読者によって本稿の教育的な面での有用性が見いだされることを期待するものである。

2.　アドバース・セレクション：部品調達問題

本稿では、伊藤の第 1 章で示されている部品調達問題について考察する。この問題はアドバース・セレクションを考える際の最も基本的な問題である。すなわち、1 台の最終生産物を作るメーカー(プリンシパル)がその生産に必要な部品をサプライヤー(エージェント)から購入することを考える。サプライヤーは自分自身の技術水準(効率性に関して、2 または 3 タイプあるとしよう)を私的情報として持っているが、メーカーには観察できない。こうした状況の下で、メーカーは自己の効用を最大にする契約を策定する。

本稿で扱うものは、タイプが離散変数のケースであり、2.1 では、サプライヤーのタイプを 2 個(効率的タイプと非効率的タイプ)とする。そこで、伊藤 1.1.2 に従って、ベンチマークのために、まず、非対称情報がないものとしてファーストベスト(最善)の均衡を解き、つぎに、伊藤 1.1.4 に従って、非対称情報があるとしてセカンドベスト(次善)の均衡を解くことにする。ファーストベスト・モデルとしては、最初に、契約を結ぼうとしているサプライヤーが効率的であることを、メーカーが知った上で契約する問題のモデルと、非効率的であると分かった上で契約する問題のモデルを個別に作成して解く。つぎに、これらの 2 つのモデルを 1 つのモデルとして解く。続いて、セカンドベスト・モデルを提示する。ここでは、サプライヤーのそれぞれが、自分以外の全てのサプライヤーのどれかであると偽る可能性があり、これを防ぐための(大域的)誘因両立制約を組み込まなければいけない。この制約のために、サプライヤーはファーストベスト以下の効用しか実現できない。

つぎはタイプが 3 個の場合である。これは、2.2 および 2.3 で検討する。サプライヤーのタイプが 2

個から一般に N 個に増えたとき、複雑になるのが誘因両立制約である。申告する可能性のあるタイプ数が多くなると、誘因両立制約の数が急速に増えることになる。そこで、追加的仮定—Spence-Mirrlees の単一交差性(single crossing property, SCP)や単調性—を加えて簡単化を図る。(SCP や単調性については、2.2.3 で詳しく説明する。)これらの仮定はモデルを簡単化して、最適契約の性質を解析的に明らかにすることを可能にする。一方、本稿では、これらの簡単化のための仮定をしない場合も考える。さらに、タイプが 3 個より多いモデルも提示する。プログラミング技術上、この種の拡張がきわめて容易であることが示される。また、この種の考察を通じて、タイプ数が増えるとこれらの簡単化のための仮定が成り立つ場合が限られてくることも示唆される。

なお、伊藤のオリジナル・モデルと本稿の数値計算モデルの間には(また、伊藤の 1.1 と 1.3 の間にも)、関数や変数に関する仮定や記号表記に違いがあるので、その相違点を表 2.1 にまとめておく。

表 2.1: 関数・変数に関する仮定の相違

関数・変数	伊藤(2003)における仮定	本稿における仮定
$b(\cdot)$	$b(0)=0$, 任意の $x<\bar{x}$ に対して $b'(x)>0$, $b'(0)=+\infty$, $b'(\bar{x})=0$ 任意の x に対して $b''(x)<0$ (ただし、$x\in X=[0,\bar{x}]$)	$b(x)=x^{0.5}$ (厳密には $b'(\bar{x})=0$ を満たさないが、近似として十分)
$c_i(\cdot)$ または $u(\cdot)$	-2 タイプ・モデル: $c_i(x_i)=\theta_i x_i$ - N タイプ・モデル($N\geq 3$): $u(x_i,\theta_i)$ は θ_i の厳密な増加関数	$c_i(x_i)=\theta_i x_i$ $u(x_i,\theta_i)=-\theta_i x_i$ ($u(x_i,\theta_i)$ は θ_i の厳密な減少関数)
タイプ i と θ_i	-2 タイプ・モデル: $0<\theta_0<\theta_1$ θ_0 が最も効率的 - N タイプ・モデル($N\geq 3$): $\theta_0<\theta_1<\ldots<\theta_N$ θ_N が最も効率的	-2 タイプ・モデル: $\begin{pmatrix}\theta_0\\ \theta_1\end{pmatrix}=\begin{pmatrix}\theta_{eff}\\ \theta_{\inf}\end{pmatrix}=\begin{pmatrix}0.2\\ 0.3\end{pmatrix}$ θ_0 が最も効率的 - N タイプ・モデル($N\geq 3$): $\begin{pmatrix}\theta_0\\ \theta_1\\ \theta_2\end{pmatrix}=\begin{pmatrix}0.1\\ 0.2\\ 0.3\end{pmatrix}$ θ_0 が最も効率的

ru（留保効用）	陽表的に考慮しない(=0)	ru を導入。ただし、本文中では $ru=0$ 。
メーカーの効用	-伊藤 1.1 $\sum_i p_i[b(x_i)-c_i(x_i)]$ $=\sum_i p_i[b(x_i)-\theta_i x_i]$ -伊藤 1.3 $S(x_i,\theta_i)-U(\theta_i)$ $=\sum_i p_i[b(x_i)-w_i]-[w_i-\theta_i x_i]$ $=\sum_i p_i[b(x_i)-\theta_i x_i]$	$\sum_i p_i[b(x_i)-c_i(x_i)]$ $=\sum_i p_i[b(x_i)-\theta_i x_i]$

2.1 2タイプ・モデル

サプライヤーのタイプを添え字 i で表し、効率的(伊藤では $i=0$ 、ここでは `eff`)と非効率的(同じく $i=1$ 、ここでは `inf`)の 2 種類があるとする。効率性は、それらの限界費用(θ_i , `theta(i)`)の高低によって決められるとする。メーカーが作る契約は、サプライヤーが自分自身のタイプを「i(効率的、または、非効率的)である」と申告するのに対して、メーカーの効用を最大化するようなメカニズム—部品の品質 x_i とサプライヤーに支払うべき価格 w_i のペア—を決めるものである。メーカーがサプライヤーのタイプを知っている(情報の非対称性なしの)場合には、サプライヤーが情報の非対称性を利用して自己に有利な申告をすることがないので、ファーストベストが実現できる。一方、メーカーがサプライヤーのタイプを知らない(情報の非対称性有りの)場合には、サプライヤーに偽りの申告をする気を起こさせない、すなわち、偽りの報告をしてもトクにならないような契約を提示する必要がある。そのような契約が、情報の非対称性に伴う非効率を少なくするという意味で最適な契約である。もちろんこの場合には、メーカーは、ファーストベストと同じ効用水準を達成できるわけではなく、セカンドベストの効用水準しか実現できない。[2]

[2] 伊藤の第 1 章では、利益と効用が互換的に用いられているが、ここでは効用だけを用いる。

2.1.1 ファーストベスト・モデル: 2 種類のサプライヤー別のモデル

部品調達問題のファーストベスト・モデル(あるいは、ベンチマーク・モデル)は、メーカーが、サプライヤーの提供する部品製造の限界費用(θ_i, `theta(i)`)を知った上で、部品の品質(x_i)とサプライヤーに支払うべき価格(w_i)を決めた契約を作るものである。ここでは、伊藤 1.1.2 に従う形で、ファーストベスト・モデルを作る。すなわち、サプライヤーが効率的(`eff`)な場合の問題と、非効率的(`inf`)な場合の問題を、別々の問題として考える。各モデルは、単純に決定変数 x_i が非負であるという条件の下での、メーカーの効用—これは非対称情報の問題がないために、総余剰に等しくなる—最大化問題である。

2.1.2 効率的サプライヤーのモデル（`PS2_F_eff.gms`）

はじめに、効率的サプライヤーのモデルの入力ファイル(`PS2_F_eff.gms`)をみていこう。このモデルは、伊藤 1.1.2 のモデルを基に、表 2.1 にまとめように一部の関数を特定化して書き換えたものである。

$$\max\ Util = \sum_i \left[b(x_i) - c_i(x_i) \right] \tag{2.1}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$c_i(x_i) - \theta_i x_i = 0 \quad \forall i \tag{2.3}$$

一般には、$i = eff, inf$ であるが、ここでは、$i = eff$ (効率的サプライヤー)のみについて問題を考えて解く。リスト 2.1 に、そのための入力ファイルを示す。そこでは、左端に行番号が付されているが、これは説明の便宜のためであって、実際には入力しない(入力するとエラーになる)。

リスト 2.1: 効率的サプライヤー・モデルの入力ファイル（`PS2_F_eff.gms`）

```
...
* Definition of Set
Set     i       type of supplier        /eff/;
* Definition of Parameters
Parameter
```

```
11        theta(i)       efficiency     /eff    0.2/;
12  * Definition of Primal/Dual Variables
13  Positive Variable
14        x(i)          quality
15        b(i)          maker's revenue
16        c(i)          cost;
17  Variable
18        Util          maker's utility;
19  Equation
20        obj           maker's utility function
21        rev(i)        maker's revenue function
22        pc(i)         participation constraint;
23  * Specification of Equations
24  obj..   Util =e= sum(i, (b(i)-c(i)));
25  rev(i)..b(i) =e= x(i)**(0.5);
26  pc(i).. c(i)-theta(i)*x(i) =e= 0;
27
28  * Setting Lower Bounds on Variables to Avoid Division by Zero
29  x.lo(i)=0.0001;;
30
31  * Defining and Solving the Model
32  Model FB1 /all/;
33  Solve FB1 maximizing Util using NLP;
34
35  Parameter
36              db(i)   derivative of b
37              w(i)    price
38  ;
39  db(i)   =0.5*x.l(i)**(-0.5);
40  w(i)    =c.l(i);
41
42  Display x.l, b.l, c.l, util.l, db, w;
43  * End of Model
```

プログラムの 8 行目にサプライヤーのタイプを表す添え字 `i` が、指示子(要は命令のようなもの)`Set` で定義されている。効率的サプライヤー(だけ)のモデルでは、効率的サプライヤーを表す `eff` だけがあれば十分なため添え字 `i` を省略しても構わないが、この後で、この同じコンピュータ・モデルを基に、非効率的サプライヤーのモデルを作ることを考えてこの命令がある。11 行目は、`eff` サプライヤーの限界費用 `theta(i)`を定義して、その値を `0.2` と設定(伊藤第 1 章にはない本稿における独自の仮定)している。

13–18 行目は、GAMS プログラム内で用いる決定変数(内生変数)の名称を定義している。理論モデ

ル(伊藤のオリジナル)における決定変数は、部品の品質 x_i だけであるが、メーカーの収入関数 $b(\cdot)$ と費用関数 $c_i(\cdot)$ を数値計算のために特定化するために、`x(i)` 以外にそれらの値 `b(i), c(i)` がコンピュータ・プログラム内の決定変数としてここで定義されている。さらに、`x(i), b(i), c(i)` は非負であるとして、`Positive Variable` という指示子で定義する。[3]もう 1 つの変数 `Util` は、これも伊藤第 1 章では陽表的に用いられていないが、このプログラムにおける目的関数の値の名称として定義する。目的関数の値の符号(値域)はモデルを解く前には分からないので、正負両方を取り得る `Variable` とする。19–22 行目は、目的関数(メーカーの効用関数)と制約式(収入関数、費用関数)の名称である。GAMS プログラム中では、制約式(目的関数を含む)名称を与えなければならない。[4]`obj` は目的関数(2.1)、`rev(i)` は収入関数(2.2)、`pc(i)` は費用関数(2.3)の名称であるとした。

24–26 行目が、最大化問題を構成する制約式(目的関数を含む)である。24 行目が目的関数である。[5]25 行目で、メーカーの収入 `b(i)` が、`x(i)` の関数として定義されている。収入関数 $b(\cdot)$ について、伊藤では一般的な形で仮定されているが、ここでは、その最も簡単な関数の代表例として $b(x_i) = x_i^{0.5}$ を仮定する。なお、GAMS では、「`=e=`」は通常の等号、(行頭以外の)「`*`」は乗算、「`**`」はベキ乗を意味する。26 行目で、伊藤 1.1.2 の通りに、サプライヤーの費用 $c_i(\cdot)$ を限界生産費 θ_i が一定の一次関数とする。

29 行目は、数値計算の途中で `x(i)` がゼロになって計算に支障が起こるのを防ぐために、`x(i)` の下限として任意の、しかし、計算結果には影響を与えない非常に小さい正数を与えるものである。もちろん、`x(i)` の解がここで与えた下限に一致してしまった場合は、この人為的な下限の設定を再考する必要が

---

[3] なお、GAMS プログラム中では、`Positive Variable` は非負の変数の意味であり、ゼロも含むことに注意。

[4] 通常であれば、「(1)式」, 「(2-3)式」といった式番号を用いるが、それと同様で、しかし、より自由度の高い式の名前を与えることができる。

[5] すべての状況 `i` に対応した収入と費用の差額の総額がメーカーの効用であるので、`i` についての和 Σ_i (GAMS プログラム中では、「`sum(i, ...)`」)をとってある。

ある。[6]32 行目は、これまでに定義されたすべて(all)の制約式(目的関数を含む)で構成されるモデルの名前を FB1 と定義し、33 行目は、このモデル FB1 を、非線形計画問題(nonlinear programming, NLP)ソルバーを使って、Util を最大化するように解くという指示である。[7]

35 行目以降では、モデルの数値解を基に、さらなる分析をするためのものである。db(i)は、関数 b(i)を x(i)の解 x.l(i)において評価した一階微分の値である。契約は、部品の品質 x(i)とともにメーカーがサプライヤーに支払う価格 w(i)を提示するものであるので、ここで w(i)を計算する(ただし、プログラムからわかるように、これは実質的に単なる代入である)。ファーストベストにおいては、契約締結時の交渉力はメーカー側にあるとしているので、w(i)はつねに生産費用 c(i)に等しい。そこで w(i)の値を c(i)の解の値に等しくしている。(モデルの解を計算に使う場合は、変数の後に「.l」(数字の 1 でなくアルファベットのエル)を付す。たとえば、x.l(i)のように。また、制約つき最大化問題のラグランジュ乗数の解を使う場合は、「.m」を付す。たとえば、pc.m(i)というように。)なお、最大化問題を構成する制約式(目的関数を含む)以外では、等号には「=e=」ではなく、普通の「=」を使う。[8]42 行目は、モデルの解(ラグランジュ乗数を含む)や上で計算した数値を表示 Display せよという指示である。

最後に、GAMS のプログラムにおいて、「*」(アスタリスク)で冒頭が始まる行は単なるメモである。プロ

[6] GAMS プログラム中で先験的に設定する変数の下限については、細江他(2004, 第 3 章)。

[7] GAMS は様々な種類の問題((非)線形計画問題、整数計画問題、相補計画問題等)を解くことができるように、それぞれの問題に応じたソルバーを備えている。プログラムのこの部分では、どのソルバーを用いて問題を解くべきかを指示している。

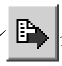
[8] ラグランジュ乗数は、最大化問題が、

$$\max\ f(x)$$

$$\text{subject to} \qquad g_i(x_i) \le 0 \qquad \forall i$$

という形式で表されるとき、そのラグランジュ乗数が非負で表現される(Varian (1992, 第 27 章))。しかし、本稿中のプログラムは、プログラム自体の解釈のしやすさを重視して、必ずしもそのような形で表されていない。したがって、ラグランジュ乗数の計算結果が負で表現される場合があるが、本質的に何ら相違はない。

グラム自体に何ら影響を与えない一方で、このメモがあることで他の人がプログラムの内容を、よりよく理解できたり、モデル作成者自身の記憶の助けになったりもする。(時間が経つと、モデル作成者自身でさえ、自分の書いたものを思い出せないことがある。)可能な限りメモを書いておくことが GAMS プログラム作成上のポイントである。

以上が入力ファイルの中身である。こうしたコンピュータ・プログラムを GAMS IDE と呼ばれるソフトウェア上で作成して、メニューバー上のアイコンクリックすると、GAMS がこのコンピュータ・プログラムを入力ファイルとして解釈して計算を行い、出力ファイル(`PS2_eff.lst`, リスト 2.2)が表示される。[9]リスト 2.2 では、左端に行番号が示されているが、入力ファイル(リスト 2.1)と同様に、説明の便宜のためである。

リスト 2.2: 効率的サプライヤー・モデルの出力ファイル(抜粋) (`PS2_eff.lst`)

```
...
  7  * Definition of Set
  8  Set     i      type of supplier       /eff/;
  9  * Definition of Parameters
 10  Parameter
 11         theta(i)       efficiency     /eff    0.2/;
 12  * Definition of Primal/Dual Variables
 13  Positive Variable
 14         x(i)          quality
...
            S O L V E     S U M M A R Y

    MODEL   FB1              OBJECTIVE  Util
    TYPE    NLP              DIRECTION  MAXIMIZE
    SOLVER  CONOPT            FROM LINE  33

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE             1.2500
...
 ** Optimal solution. Reduced gradient less than tolerance.
```

[9] GAMS および GAMS IDE の利用方法については、細江他(2004, 第 3 章および付録 C)、または、Brooke et al. (2010)、および、より技術的な点は McCarl (2009)参照。

```
...
                         LOWER     LEVEL     UPPER    MARGINAL

---- EQU obj              .         .         .        1.000

  obj  maker's utility function

---- EQU rev  maker's revenue function

       LOWER     LEVEL     UPPER    MARGINAL

eff      .         .         .       1.000

---- EQU pc  participation constraint

       LOWER     LEVEL     UPPER    MARGINAL

eff      .         .         .      -1.000

---- VAR x  quality

       LOWER     LEVEL     UPPER    MARGINAL

eff 1.0000E-7    6.250     +INF      EPS

---- VAR b  maker's revenue

       LOWER     LEVEL     UPPER    MARGINAL

eff      .       2.500     +INF       .

---- VAR c  cost

       LOWER     LEVEL     UPPER    MARGINAL

eff      .       1.250     +INF       .

                         LOWER     LEVEL     UPPER    MARGINAL

---- VAR Util           -INF      1.250     +INF       .
...
----     42 VARIABLE x.L  quality

eff 6.250


----     42 VARIABLE b.L  maker's revenue

```

```
eff 2.500


----      42 VARIABLE c.L  cost

eff 1.250


----      42 VARIABLE Util.L          =    1.250  maker's utility

----      42 PARAMETER db  derivative of b

eff 0.200


----      42 PARAMETER w  price

eff 1.250
...
```

この出力ファイルを解釈する。出力ファイルの前半部には、入力ファイルの中身が行番号つきでそのまま表われる。その先にある SOLVE SUMMARY に、「** Optimal solution」の表示があることを確認する。(この表示がない場合には、以下にどのような数値が表されていても、解としては意味を持たない。)GAMS の解の出力には EQU(おもに制約式のラグランジュ乗数の値を表示する)と VAR(おもに決定変数(内生変数)の値を表示する)の 2 種類がある。はじめに EQU ブロックが表示される。EQU ブロックの MARGINAL の列に、それぞれの式のラグランジュ乗数が示されている。たとえば、制約式 rev(i)のラグランジュ乗数の値は 1.000 である。(また、どのような最適化問題であっても、目的関数に設定された制約式 obj のラグランジュ乗数の値はつねに 1 であるから議論しない。)次の VAR ブロックに、決定変数(内生変数)、および目的関数の最適解がある。

このモデルの解、および、それに基づいて計算された変数の値は、伊藤 1.1.2 の説明と合致する(表 2.2)。部品の品質の解 x.l(i)における、メーカーの限界収入 db(i)は、サプライヤーi の限界費用 theta(i)に等しい。take-it-or-leave-it offer を考えて、サプライヤーがメーカーから受け取る価格 w(i)が費用 c(i)に等しいという条件 pc(i)を課しているので、サプライヤーの効用はゼロとなる。したがって、このメーカーの効用最大化問題を総余剰最大化問題と呼んでも構わない。

表 2.2: 効率的サプライヤー・モデルの数値解

変数	プログラム中での変数名	数値解
部品の品質	`x(i)`	6.250
メーカーの収入	`b(i)`	2.500
メーカーの費用	`c(i)`	1.250
解(`x.l(i)`)で評価した b_i の一階の微係数	`db(i)`	0.200
メーカーの支払い価格	`w(i)`	1.250
メーカーの効用	`Util`	1.250

2.1.3 非効率的サプライヤーのモデル（`PS2_F_inf.gms`）

非効率的サプライヤーのモデルの入力ファイル(`PS2_F_inf.gms`)をみていこう。問題としては、(2.1)–(2.3)の最大化問題を $i = inf$ のみについて解けばよい。そのために非効率的サプライヤーのコンピュータ・プログラムは、効率的サプライヤーのプログラム(リスト 2.1)の 2 箇所を変更するだけでよい。

・8 行目の指示子 `set` で定義しているタイプ集合 `i` の要素を「`inf`」に書き替える。

・11 行目の `theta(i)` の値を、非効率的サプライヤーの限界費用としてここで仮定する「`0.3`」に書き替える。

このモデルの解、および、それに基づいて計算された値は表 2.3 の通りである。部品の品質の解 `x.l(i)` における、メーカーの限界収入 `db(i)` がサプライヤーの限界費用 `theta(i)` に一致することや、メーカーの効用最大化問題として定式化されたこの問題が実質的に総余剰最大化問題であることなどは、2.1.2 で指摘したことと同じである。

表 2.3: 非効率的サプライヤー・モデルの数値解

変数	プログラム中での変数名	数値解
部品の品質	`x(i)`	2.778
メーカーの収入	`b(i)`	1.667
メーカーの費用	`c(i)`	0.833
解(`x.l(i)`)で評価した b_i の一階の微係数	`db(i)`	0.300
メーカーの支払い価格	`w(i)`	0.833
メーカーの効用	`Util`	0.833

2.1.4　両サプライヤーを統合したファーストベスト・モデル（`PS2_F.gms`）

これまでに構築してきた 2 つのモデルを基に、効率的サプライヤーと非効率的サプライヤーの 2 種類が、ある確率 p_i で存在し、かつ、彼がどちらのタイプであるかがメーカーにとっても明らかであるとする状況の下で達成できるファーストベストの最適契約を考える。すなわち、問題(2.1)–(2.3)とほぼ同じものを両方のタイプのサプライヤー($i = eff, inf$)が現れることを前提にして解けばよい。

$$\max\ Util = \sum_i p_i \left[b(x_i) - w_i \right] \tag{2.1'}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$w_i - \theta_i x_i \geq ru \qquad \forall i \tag{2.3'}$$

前項のサプライヤーのタイプ別モデルとは目的関数(2.1’)が異なっている。ここではタイプ i の事前確率 p_i で加重したメーカーの効用を最大化するようになっている。また、これまでのモデルでは、take-it-or-leave-it offer を先験的に仮定して w_i を $c_i(x_i)$ に置き換えて問題を簡単化していたが、ここではそのようなことは仮定せずにそのままサプライヤー i への支払い価格 w_i を用いる。なお、支払い価格 w_i がサプライヤーの効用 $\theta_i x_i$ と留保効用 ru の和を上回る可能性を考えて(この点はつぎのセカンドベストの状況で重要となる)、(2.3’)では不等号を用いる。

コンピュータ・プログラム上で、前項のものと異なる点を中心に説明する(リスト 2.3)。8 行目で、タイプを表す添え字の集合 `i` の要素として効率的サプライヤー`eff` と非効率的サプライヤー`inf` の両方を入れる。11–16 行目で、それぞれのタイプのサプライヤーの限界費用 `theta(i)`の値と、サプライヤーが効率的 `eff` あるいは非効率的 `inf` である確率 `p(i)`を仮定する。17 行目で留保効用 `ru` を与える。この数値モデルでは、伊藤にならってゼロに設定しているが、必要に応じて、ゼロ以外の適当な数値を与えることができる。(前項のモデルでも同様。)

20–25 行目では、この最大化問題の決定変数を定義する。前項とは違って、費用 `c(i)`ではなく、メーカーの支払い価格 `w(i)`を決定変数とする。26–29 行目は、目的関数と制約式の名前を定義してい

る。32–34 行目が最大化問題を構成する一連の制約式である。32 行目が目的関数 `obj` である。メーカーの効用 `Util` は、各サプライヤーの作る部品から得られる利益(`b(i)`と `w(i)`の差額)に、効率的サプライヤーと非効率サプライヤーが出現する確率 `p(i)`を掛けて合算したものである。33 行目は、それぞれのサプライヤーからの収入関数 `b(i)`の定義式 `rev(i)`であり、前項で解説したコンピュータ・モデルと同じ関数 $b_i = x_i^{0.5}$ を仮定する。34 行目は、参加制約条件 `pc(i)`である。不等号(≥)は、プログラム中では「`=g=`」と表す。40–41 行目ではモデル名を定義して、メーカーの期待効用 `Util` を最大化するようにモデルを解くことを指示している。

リスト 2.3: 両サプライヤーを統合したファーストベスト・モデルの入力ファイル（`PS2_F.gms`）

```
...
 7 * Definition of Set
 8 Set     i      type of supplier       /eff, inf/;
 9
10 * Definition of Parameters
11 Parameter
12        theta(i)       efficiency      /eff    0.2
13                                        inf     0.3/
14        p(i)           probability of type
15                                       /eff    0.2
16                                       inf     0.8/;
17 Scalar  ru  reservation utility        /0/;
18
19 * Definition of Primal/Dual Variables
20 Positive Variable
21        x(i)          quality
22        b(i)          maker's revenue
23        w(i)          price;
24 Variable
25        Util          maker’s utility;
26 Equation
27        obj           supplier's profit function
28        rev(i)         maker's revenue function
29        pc(i)         participation constraint;
30
31 * Specification of Equations
32 obj..   Util =e= sum(i, p(i)*(b(i)-w(i)));
33 rev(i)..b(i) =e= x(i)**(0.5);
34 pc(i).. w(i)-theta(i)*x(i) =e= ru;
35
```

```
* Setting Lower Bounds on Variables to Avoid Division by Zero
x.lo(i)=0.0001;;

* Defining and Solving the Model
Model FB1 /all/;
Solve FB1 maximizing Util using NLP;

* End of Model
```

以上が入力ファイルである。 続いて、この出力ファイル(`PS2_F.lst`)を確認する(リスト 2.4)。まず、`EQU` のブロックでは、制約式 `pc(i)`のラグランジュ乗数(128–133 行目の `MARGINAL` 値)が、効率的サプライヤー`eff`、非効率的サプライヤー`inf` ともに、ゼロではないので、この制約が有効であることが分かる。すなわち制約は等号で成立しており(メーカーからサプライヤーiへの支払い価格w_iがそれぞれの費用$\theta_i x_i$に厳密に等しい)、したがってサプライヤーの効用水準がゼロである(正確には留保効用水準に一致する)ことがわかる。`VAR` ブロックでは、決定変数である品質 `x(i)`、価格 `w(i)`、メーカーの収入 `b(i)`、およびメーカーの効用 `Util` の解が示されている。

リスト 2.4: 両サプライヤーを統合したファーストベスト・モデルの出力ファイル(抜粋) (`PS2_F.lst`)

```
...
 74              S O L V E     S U M M A R Y
 75
 76     MODEL   FB1               OBJECTIVE  Util
 77     TYPE    NLP               DIRECTION  MAXIMIZE
 78     SOLVER  CONOPT             FROM LINE  41
 79
 80 **** SOLVER STATUS     1 NORMAL COMPLETION
 81 **** MODEL STATUS      2 LOCALLY OPTIMAL
 82 **** OBJECTIVE VALUE               0.9167
...
115                     LOWER     LEVEL     UPPER    MARGINAL
116
117 ---- EQU obj           .         .         .        1.000
118
119   obj  supplier's profit function
120
121 ---- EQU rev  maker's revenue function
122
123       LOWER     LEVEL     UPPER    MARGINAL
```

```

eff      .        .        .       0.200
inf      .        .        .       0.800

---- EQU pc  participation constraint

      LOWER     LEVEL     UPPER    MARGINAL

eff      .        .        .      -0.200
inf      .        .        .      -0.800

---- VAR x  quality

      LOWER     LEVEL     UPPER    MARGINAL

eff 1.0000E-7   6.250     +INF  2.0951E-9
inf 1.0000E-7   2.778     +INF      EPS

---- VAR b  maker's revenue

      LOWER     LEVEL     UPPER    MARGINAL

eff      .       2.500     +INF      .
inf      .       1.667     +INF      .

---- VAR w  price

      LOWER     LEVEL     UPPER    MARGINAL

eff      .       1.250     +INF      .
inf      .       0.833     +INF      .

                    LOWER     LEVEL     UPPER    MARGINAL

---- VAR Util       -INF      0.917     +INF      .
```

このモデルの解を表 2.4 にまとめた。メーカーの効用以外はどの値も、前項で効率的サプライヤーと非効率的サプライヤーの問題を別々にモデル化したときの解(表 2.2–2.3)と一致している。

表 2.4: 両サプライヤーを統合したファーストベスト・モデルの数値解

変数	プログラム中での変数名	サプライヤーのタイプ	数値解
部品の品質	`x(i)`	効率的(`eff`)	6.250
		非効率的(`inf`)	2.778
メーカーの収入	`b(i)`	効率的(`eff`)	2.500
		非効率的(`inf`)	1.667
メーカーの支払い価格	`w(i)`	効率的(`eff`)	1.250
		非効率的(`inf`)	0.833
メーカーの効用	`Util`		0.917

2.1.5 両サプライヤーを統合したセカンドベスト・モデル（`PS2_S.gms`）

部品調達問題のセカンドベスト・モデルは、メーカーが、サプライヤーのタイプ、すなわち、サプライヤーの提供する部品製造の限界費用 θ_i を知らないままに、部品の品質 x_i とサプライヤーに支払うべき価格 w_i を含む契約を作るものである。効率的サプライヤーが限界費用を真実のものより高く申告することで(すなわち非効率的サプライヤーの振りをして)、その虚偽の限界費用に応じて過大に支払われる価格と真実の費用との間の差額を懐に入れる可能性がある。メーカーはそれを防ぐために、タイプを偽って申告してもトクにならない条件—誘因両立制約—をあらたに考慮して最適契約を考えることになる。

セカンドベスト・モデルは、伊藤 1.1.4 の問題(p)にそのまま従う。これは、本稿 2.1.4 のファーストベストの問題(2.1'), (2.2), (2.3')に、誘因両立制約(2.4)を付加することで作ることができる。なお、ファーストベスト・モデルは 2.1.2 や 2.1.3 のように効率的サプライヤーのモデルと非効率的サプライヤーのモデルを別々に作ることができるが、セカンドベスト・モデルは両サプライヤーを統合したモデルでなければならない。それは、誘因両立制約に、両方のサプライヤーの品質 `x(i)`とメーカーの支払い `w(i)`を、決定変数として入れなければならないからである。

セカンドベスト・モデルは以下の通りである。

$$\max\ Util = \sum_i p_i\left[b(x_i) - w_i\right] \tag{2.1'}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$w_i - \theta_i x_i \geq ru \qquad \forall i \tag{2.3'}$$

$$w_i - \theta_i x_i \geq w_j - \theta_i x_j \qquad \forall i \neq j \tag{2.4}$$

θ_i の限界費用を持つタイプ i のサプライヤーにとって、(2.4)の左辺は自らのタイプを正直に申告したときに提示される価格 w_i と品質 x_i から得られる効用であり、その右辺は、自らのタイプを $j(\neq i)$ と偽って申告したときに提示される価格 w_j と品質 x_j から得られる効用である。左辺が右辺を下回らないようにすることで、自らのタイプを正直に申告させることができる。リスト 2.5 では、タイプの記号を i と同じ意味で j も用いるために、9 行目で `Alias` 指示子を使って、`j` が `i` の別名(alias)であり、互換的に用いる旨を宣言している。

この(タイプ `i` のサプライヤーが自らのタイプについて「タイプ `j` である」と申告することを防ぐ)誘因両立制約(2.4)の名称 `ic(i,j)` は 30 行目で定義され、その制約式自体は 36 行目で記述される。ところで、$i = eff, inf$ についてプログラム中の制約式 `ic(i,j)` に相当するものの内容を書き下してみると、

$$w_{eff} - \theta_{eff} x_{eff} \geq w_{inf} - \theta_{eff} x_{inf} \tag{2.4a}$$

$$w_{eff} - \theta_{eff} x_{eff} \geq w_{eff} - \theta_{eff} x_{eff} \tag{2.4b}$$

$$w_{inf} - \theta_{inf} x_{inf} \geq w_{eff} - \theta_{inf} x_{eff} \tag{2.4c}$$

$$w_{inf} - \theta_{inf} x_{inf} \geq w_{inf} - \theta_{inf} x_{inf} \tag{2.4d}$$

の 4 本の制約を課していることがわかる。すなわち、このプログラム中の制約式は、本来 2 本しかない制約式(2.4)に比べて、(2.4b)と(2.4d)の 2 本の冗長な制約式を含んでいる。しかしながら、これら 2 本の冗長な制約式は、一見してすぐにわかるように右辺と左辺が同じものであるから常に等号で成立する自明なものである。したがって、解いている問題に本質的な違いはない。[10]プログラムの他の部分は、両サプライヤーを統合した前項のファーストベスト・モデル(`PS2_F.gms`)と変わらない。

---

[10] もしこの冗長さが気になるのであれば、プログラム中の制約式を、

```
ic(i,j)$(ord(i) ne ord(j))..w(i)-theta(i)*x(i) =g= w(j)-theta(i)*x(j);
```

と書き換えればよい。ここで「`$(…)`」は括弧内を条件とするダミー変数であり、「`ord(i)`」は添え字 `i` が、それが定義された集合の中で何番目に現れるのか、その順番を表す関数であり、「`ne`」は≠の意味である。

リスト 2.5: セカンドベスト・モデルの入力ファイル (`PS2_S.gms`)

```
...
* Definition of Set
Set     i       type of supplier       /eff, inf/;
Alias (i,j);
* Definition of Parameters
Parameter
       theta(i)       efficiency      /eff    0.2
                                       inf     0.3/
       p(i)           probability of type
                                       /eff    0.2
                                       inf     0.8/;
Scalar  ru            reservation utility    /0/;

* Definition of Primal/Dual Variables
Positive Variable
       x(i)          quality
       b(i)          maker's revenue
       w(i)          price;
Variable
       Util          maker's utility;
Equation
       obj           maker’s utility function
       rev(i)        maker's revenue function
       pc(i)         participation constraint
       ic(i,j)       incentive compatibility constraint;

* Specification of Equations
obj..   Util =e= sum(i, p(i)*(b(i)-w(i)));
rev(i)..b(i) =e= x(i)**(0.5);
pc(i).. w(i)-theta(i)*x(i) =g= ru;
ic(i,j)..w(i)-theta(i)*x(i) =g= w(j)-theta(i)*x(j);

* Setting Lower Bounds on Variables to Avoid Division by Zero
x.lo(i)=0.0001;;

* Defining and Solving the Model
Model SB1 /all/;
Solve SB1 maximizing Util using NLP;

* End of Model
```

この入力ファイルを基に問題を解いた結果は、リスト 2.6 に示されている。

リスト 2.6: セカンドベスト・モデルの出力ファイル(抜粋) (PS2_S.lst)

```
...
---- EQU rev  maker's revenue function

       LOWER     LEVEL     UPPER    MARGINAL

eff      .         .         .       0.200
inf      .         .         .       0.800

---- EQU pc  participation constraint

       LOWER     LEVEL     UPPER    MARGINAL

eff      .       0.237     +INF       .
inf      .         .       +INF    -1.000

---- EQU ic  incentive compatibility constraint

           LOWER     LEVEL     UPPER    MARGINAL

eff.inf      .         .       +INF    -0.200
inf.eff      .       0.388     +INF      EPS

---- VAR x  quality

       LOWER     LEVEL     UPPER    MARGINAL

eff 1.0000E-7    6.250     +INF  6.7551E-8
inf 1.0000E-7    2.367     +INF       .

---- VAR b  maker's revenue

       LOWER     LEVEL     UPPER    MARGINAL

eff      .       2.500     +INF       .
inf      .       1.538     +INF       .

---- VAR w  price

       LOWER     LEVEL     UPPER    MARGINAL

eff      .       1.487     +INF       .
inf      .       0.710     +INF       .

                     LOWER     LEVEL     UPPER    MARGINAL

---- VAR Util         -INF     0.865     +INF       .
```

. . .	

2.1.6 ファーストベスト・モデルとセカンドベスト・モデルの解の比較

ファーストベストとセカンドベストを比較した表 2.5 の結果は次のようにまとめられ、伊藤 1.1.4 の解説と合致することはこれまでと同様である。すなわち、誘因両立制約は、効率的サプライヤー`eff` についてだけが有効である(この制約のラグランジュ乗数がゼロでない)。すなわち、自らのタイプを偽ったときと、正直に申告したときの間で無差別となっている。メーカーは、効率的サプライヤーに虚偽の申告をさせないようにするために、*情報の非対称性がなかったときと比べて*、より高い価格 w_{eff} を「効率的」と申告したものに提示する。その一方で、(単純に、「効率的」と申告したものへの提示価格 w_{eff} を引き上げるだけでは、効率的サプライヤーが得る情報レントが大きくなりすぎるので、w_{eff} の w_{inf} に対する相対的な価格 w_{eff} / w_{inf} を引き上げるように)、より低い価格 w_{inf} を「非効率的」と申告したものに提示している。すなわち、サプライヤーへの支払い価格 `w(i)`は、セカンドベストでは、ファーストベストに比べ、効率的サプライヤー`eff` で増加し、非効率的サプライヤー`inf` で減少する。この結果、効率的サプライヤーには、参加制約 `pc(i)`の両辺の差額(スラック, slack)0.237—これはリスト 2.6 の 134 行目、`EQU` ブロックにある `pc("eff")`の `LEVEL` 値と `LOWER` 値の差として表される—に等しい情報レントが発生している。[11]現在のモデルでは留保効用をゼロと仮定しているので、情報レントが効率的サプライヤーの効用となる。

[11] `EQU` ブロックの `LOWER`, `LEVEL`, `UPPER` 値それぞれが何を示しているかについては McCarl(2009, 第 2.4.5.2 項)参照。なお、`Solve` 命令の前(のどこでもよい)に「`Option solslack=1;`」というオプションを記述しておくと、出力ファイルの `EQU` ブロックには、`LEVEL` の代わりに `SLACK` がそのまま出力される。

表 2.5: ファーストベスト・モデルとセカンドベスト・モデルの解の比較

変数・制約	プログラム中での変数名	サプライヤーのタイプ	セカンドベスト・モデルの解	ファーストベスト・モデルの解*
部品の品質	`x(i)`	効率的(`eff`)	6.250	6.250
		非効率的(`inf`)	2.367	2.778
メーカーの収入	`b(i)`	効率的(`eff`)	2.500	2.500
		非効率的(`inf`)	1.538	1.667
メーカーの支払い価格	`w(i)`	効率的(`eff`)	1.487	1.250
		非効率的(`inf`)	0.710	0.833
メーカーの効用	`Util`		0.865	0.917
サプライヤーの効用(=情報レント)	`pc.l(i) -pc.lo(i)`	効率的(`eff`)	0.237	0.000
		非効率的(`inf`)	0.000	0.000
誘因両立制約のラグランジュ乗数	`ic("eff","inf")` 効率的が非効率的を偽る		–0.200	–
	`ic("inf","eff")` 非効率的が効率的を偽る		0.000	–
参加制約のラグランジュ乗数	`pc.m(i)`	効率的(`eff`)	0.000	–0.200
		非効率的(`inf`)	–0.100	–0.800

注: ファーストベスト・モデルの数値解の一部は、表 2.4 に示したものと同じものである。

一方、非効率的サプライヤーには、誘因両立制約が有効ではない(=不等号で成立)ことからわかるように、虚偽の申告しても効用が増加しないために、自らのタイプを偽る誘因はない。その結果、非効率的サプライヤーの参加制約が有効になり、すべての情報レントはメーカーに吸い上げられてしまって、得られる効用は留保効用水準(これはゼロと仮定されている)だけにとどまる。なお、ファーストベストにおいては、両サプライヤーについて自らのタイプに関する情報はメーカーに完全に見透かされているから情報レントは発生せず、どちらの参加制約も有効であった。

品質 `x(i)` については、効率的サプライヤー`eff` のセカンドベスト解は、ファーストベスト解に等しい。これに対して、非効率的サプライヤー`inf` のセカンドベスト解は、上で議論したように、効率的サプライヤーに虚偽の申告をさせないように非効率的サプライヤーに対する提示価格を引き下げたために、ファーストベスト解に比べ過小になる。また、メーカーの期待効用 `Util` は、ファーストベストに比べて、セカンドベストでは減少する。この減少は、効率的サプライヤーが留保効用水準を上回る効用水準を情報レ

ントとして 0.237 だけ得るようになったことと、非効率的サプライヤーへの提示価格が低下して品質が非効率的になったこと(上の数値問題では x_{inf} が 0.411 だけ低下した)のためである。

2.1.7　ファーストベストとセカンドベストを同時に解く（`PS2_F&S.gms`）

これまでは、2 つの入力ファイルを作成し、それぞれの中でファーストベスト・モデル(`PS2_F.gms`)とセカンドベスト・モデル(`PS2_S.gms`)を別々に解いていた。両者を比較するとこれらの違いは誘因両立制約 `ic(i,j)`の有無のみである。そこで、1 つのプログラムの中で、これら 2 つの問題を同時に解くプログラムを紹介する(リスト 2.7)。プログラムの 39 行目まではセカンドベスト・モデル(リスト 2.5)とまったく同じである。42 行目で `obj`, `rev(i)`, `pc(i)`の 3 つの制約式(目的関数となるべきものも含む、以下同じ)から構成されるファーストベスト・モデルを `FB2` と名付け、これを 43 行目で `Util` を最大化するようにしている。45 行目では、`obj`, `rev(i)`, `pc(i)`、および、`ic(i,j)`の 4 つの制約式から構成されるセカンドベスト・モデルを `SB2` と名付け、これを 46 行目で最大化問題として解くようにしている。

このモデルに関してプログラム技術上で注意するべき点は、2 点ある。これまでのプログラムでは、`Model` 指示子でモデル名を定義する際には、(その行よりも前の行で)定義したすべての制約式を用いてモデルを構成することを考えていたので、

```
Model　モデル名　　　/all/;
```

と記述していた。しかし、より柔軟に、どの制約式を用いてモデルを構築するかを選択することもできる。その際には、

```
Model　モデル名　　　/制約式1, 制約式2, …/;
```

というように、モデルに含むべき制約式を(コンマで区切って)列挙する。ただし、制約式に添え字がついていても、そこでは添え字をつけない。

リスト 2.7: ファーストベスト・モデルとセカンドベスト・モデルを同時に解く入力ファイル (PS2_F&S.gms)

```
...
* Definition of Set
Set    i      type of supplier       /eff, inf/;
Alias (i,j);
* Definition of Parameters
Parameter
       theta(i)       efficiency     /eff    0.2
                                      inf     0.3/
       p(i)           probability of type
                                      /eff    0.2
                                      inf     0.8/;
Scalar  ru            reservation utility    /0/;

* Definition of Primal/Dual Variables
Positive Variable
       x(i)          quality
       b(i)          maker's revenue
       w(i)          price;
Variable
       Util          maker's utility;
Equation
       obj           maker's utility function
       rev(i)        maker's revenue function
       pc(i)         participation constraint
       ic(i,j)       incentive compatibility constraint;

* Specification of Equations
obj..   Util =e= sum(i, p(i)*(b(i)-w(i)));
rev(i)..b(i) =e= x(i)**(0.5);
pc(i).. w(i)-theta(i)*x(i) =g= ru;
ic(i,j)..w(i)-theta(i)*x(i) =g= w(j)-theta(i)*x(j);

* Setting Lower Bounds on Variables to Avoid Division by Zero
x.lo(i)=0.0001;;

* Defining and Solving the Model
Model FB1 /obj,rev,pc/;
Solve FB1 maximizing Util using NLP;

Model SB1 /obj,rev,pc,ic/;
Solve SB1 maximizing Util using NLP;

* End of Model
```

いまひとつは、1 つの入力ファイルの中で複数の `Solve` 命令を書くことで複数のモデルを解くことができ、そのとき、出力ファイルには `Solve` 命令の順番通りに同じ数だけの `SOLVE SUMMARY` が出力されることである。このモデルの出力ファイル(`PS2_S.lst`)では、それぞれの `SOLVE SUMMARY` の冒頭にモデル名(81 行目に `FB1`、197 行目に `SB1`)が示される(リスト 2.8)。これらの解が、ファーストベスト・モデル(`PS2_F.gms`)とセカンドベスト・モデル(`PS2_S.gms`)を別々に解いた結果(表 2.5)と一致することも確認できるであろう。

リスト 2.8: ファーストベスト・モデルとセカンドベスト・モデルを同時に解いた出力ファイル(抜粋)(`PS2_F&S.lst`)

```
...
 79              S O L V E      S U M M A R Y
 80
 81     MODEL   FB1               OBJECTIVE  Util
 82     TYPE    NLP               DIRECTION  MAXIMIZE
 83     SOLVER  CONOPT             FROM LINE  43
 84
 85 **** SOLVER STATUS     1 NORMAL COMPLETION
 86 **** MODEL STATUS      2 LOCALLY OPTIMAL
 87 **** OBJECTIVE VALUE                0.9167
...
195              S O L V E      S U M M A R Y
196
197     MODEL   SB1               OBJECTIVE  Util
198     TYPE    NLP               DIRECTION  MAXIMIZE
199     SOLVER  CONOPT             FROM LINE  46
200
201 **** SOLVER STATUS     1 NORMAL COMPLETION
202 **** MODEL STATUS      2 LOCALLY OPTIMAL
203 **** OBJECTIVE VALUE                0.8654
...
```

2.2 3 タイプ・モデル

2.2.1 モデルの概要と 2 タイプ・モデルとの相違点

前節では、伊藤 1.1「例 部品調達の問題」に従い、効率的と非効率的の 2 タイプ・モデルを作った。これに対して、ここでは伊藤 1.3「分析 タイプが離散変数のケース」の応用として、3 タイプ(0, 1, 2 であ

り、小さい数字であるほど効率的)をもつモデルを作る。

2 タイプ・モデルとNタイプ・モデルの間で、大きく異なってくるのは、セカンドベスト・モデルにおける誘因両立制約である。2 タイプ・モデルでは、eff が inf と偽ることと、inf が eff と偽ることの 2 つを防止する条件を課せばすべての場合を網羅できたが、タイプがN個の場合は、同様に「総当たり」すなわち、各サプライヤーにとって自分以外の全てのサプライヤーとの組み合わせを考えると、全部で$(N-1)N$本の誘因両立制約が必要になる。たとえば、3 タイプであれば 6 本で済むが、10 タイプであれば 90 本にものぼる。

この「総当たり」に対応した誘因両立制約を大域的条件(または大域的誘因両立制約)と呼んでいる。2 タイプ・モデルではあるが、この制約を課したものが伊藤 1.1.4 の問題(p)である。これに対して、伊藤 1.3 では、追加的な条件(Spence-Mirrlees の単一交差性、以下 SCP)を仮定して、大域的な誘因両立制約を、より簡単な条件—これを局所的条件(または局所的誘因両立制約と呼ぶ)—に置き換えて問題を簡単化して解いている。(SCP については、セカンドベスト・モデルのところで詳しく説明する。)こうして作られたものが、伊藤 1.3.2 の問題(P_N)である。さらにこの局所的条件も、解の単調性を仮定することにより、本来 2 本で構成される局所的条件のうちの 1 本のみを考えることで済む。これに従い、次の問題(P'_N)では、単調性制約(M_N)を外して解いた上で、解が単調性を満たすことを示すという方法を取っている。本稿ではまずこの簡単化した問題を解くことにする。

こうした伊藤の議論に従い、2.2.2 でファーストベスト・ベストモデル(ここでは誘因両立制約が必要ないので、SCP や単調性は無関係)を作った後で、2.2.3 ではそのファーストベスト・モデルを基に、SCP と単調性が成立するとして、局所的条件を使ったセカンドベスト・モデルを、次いで、2.2.4 では同じモデルで、局所的条件を大域的条件に置き換えたものを解き、両者が同一の解を得ることを示す。SCP が成立していないモデルや単調性が成立していないモデルは、次節のモデルの拡張のところで説明する。

なお、伊藤 1.1 と 1.3 の間で記号表現や関数の定義が異なっている点がある。本稿では伊藤 1.1 の表現に従うものとしているので、伊藤 1.3 と 2 つの点で表現が異なる。第 1 は記号表現上の違いである。伊藤 1.3 では、限界費用を描写するパラメータθ_iのタイプiに関する順序づけが、添え字の数字が大き

いほど効率的なサプライヤーということに変わっている(表 2.1)。しかし本稿では一貫して伊藤 1.1 に従い、タイプ 0 が最も効率的であるとする。この結果いくつかの部分で伊藤 1.3 の記述を読み替える必要があり、適宜指摘する。第 2 は目的関数の表現上の違いである。伊藤 1.3 では、最終的に目的関数や制約式を、総余剰 S やサプライヤーの効用 U を用いて表現しなおした問題(P_N'')を用いて議論している。しかし、本稿ではこれまで通りに、$\sum_i p_i\left[b(x_i)-w_i\right]$を最大化し、制約条件もこれまで通り x_i と w_i の関係として記述する。

2.2.2 ファーストベスト・モデル（PS3_F.gms）

3 タイプのファーストベスト・モデルは、本稿 2.1.4 の 2 タイプのファーストベスト・モデルを単純に拡張すればよい(リスト 2.9)。8 行目で 3 つのタイプを定義している点、12–14 行目で theta(i)をタイプ 0 が最も効率的(限界費用が低い)になるように定義・設定している。そのほか、モデルの名前を 42 行目で(FB1 ではなく)FB2 とした点が異なっている。

リスト 2.9: 3 タイプのファーストベスト・モデルのための入力ファイル（PS3_F.gms）

```
...
  7 | * Definition of Set
  8 | Set     i       type of supplier       /0,1,2/;
  9 | Alias (i,j);
 10 | * Definition of Parameters
 11 | Parameter
 12 |        theta(i)       efficiency     /0      0.1
 13 |                                       1      0.2
 14 |                                       2      0.3/
 15 |        p(i)           probability of type
 16 |                                      /0      0.2
 17 |                                       1      0.5
 18 |                                       2      0.3/;
 19 | Scalar  ru            reservation utility    /0/;
 20 |
 21 | * Definition of Primal/Dual Variables
 22 | Positive Variable
 23 |        x(i)          quality
```

```
      b(i)          maker's revenue
      w(i)          price;
Variable
      Util          maker's utility;
Equation
      obj           maker’s utility function
      rev(i)        maker's revenue function
      pc(i)         participation constraint

* Specification of Equations
obj..  Util =e= sum(i, p(i)*(b(i)-w(i)));
rev(i)..b(i) =e= x(i)**(0.5);
pc(i).. w(i)-theta(i)*x(i) =g= ru;

* Setting Lower Bounds on Variables to Avoid Division by Zero
x.lo(i)=0.0001;;

* Defining and Solving the Model
Model FB2 /all/;
Solve FB2 maximizing Util using NLP;

* End of Model
```

2.2.3 局所的誘因両立制約を用いたセカンドベスト・モデル（`PS3_S.gms`）

タイプが 2 個のモデルから 3 個のモデルに拡張されることで複雑になる(大局的)誘因両立制約を、SCP の仮定を用いて、より簡単な局所的条件に置き換えたり、単調性を仮定したりして、モデルを簡単化できることは先に述べたとおりである。本項のセカンドベスト・モデルと前項のファーストベスト・モデルとの違いは、局所的誘因両立制約が加わっている 1 点のみである。

局所的条件の詳細な説明に入る前に、以下では、N タイプ・モデルの誘因両立制約を一般的に検討することにする。一般にタイプ i が自らのタイプをどのように表明するかを考えると、表 2.6 の薄い灰色と濃い灰色の部分すべてについて可能性があるので、大域的誘因両立制約を考えなければならない。

表 2.6: 大域的誘因両立制約と局所的誘因両立制約

		表明するタイプ						
		0	$\cdots$	$i-1$	i	$i+1$	$\cdots$	$N-1$
真のタイプ	0	$U(\theta_0 \mid \theta_0)$	$\cdots$	$U(\theta_{i-1} \mid \theta_0)$	$U(\theta_i \mid \theta_0)$	$U(\theta_{i+1} \mid \theta_0)$	$\cdots$	$U(\theta_{N-1} \mid \theta_0)$
	$\cdots$				$\cdots$			
	$i-1$				$U(\theta_i \mid \theta_{i-1})$			
	i	$U(\theta_0 \mid \theta_i)$	$\cdots$	$U(\theta_{i-1} \mid \theta_i)$	$U(\theta_i \mid \theta_i)$	$U(\theta_{i+1} \mid \theta_i)$	$\cdots$	$U(\theta_{N-1} \mid \theta_i)$
	$i+1$				$U(\theta_i \mid \theta_{i+1})$			
	$\cdots$				$\cdots$			
	$N-1$				$U(\theta_i \mid \theta_{N-1})$			

このとき、タイプ i に正直に自らのタイプを申告させるための条件は(タイプ i がタイプ j であると申告したときの効用を $U(\theta_j \mid \theta_i)$ として)、

$$U(\theta_i \mid \theta_i) \geq U(\theta_j \mid \theta_i) \qquad \forall j$$

となる。書き下すと、大域的誘因両立制約は、

$$U(\theta_i \mid \theta_i) \geq U(\theta_0 \mid \theta_i)$$

$$U(\theta_i \mid \theta_i) \geq U(\theta_1 \mid \theta_i)$$

$$U(\theta_i \mid \theta_i) \geq U(\theta_2 \mid \theta_i)$$

$$\cdots$$

$$U(\theta_i \mid \theta_i) \geq U(\theta_{i-1} \mid \theta_i)$$

$U(\theta_i \mid \theta_i) \geq U(\theta_i \mid \theta_i)$ (ただしこれは自明な条件)

$$U(\theta_i \mid \theta_i) \geq U(\theta_{i+1} \mid \theta_i)$$

$$\cdots$$

$$U(\theta_i \mid \theta_i) \geq U(\theta_{N-1} \mid \theta_i)$$

となり数が極端に多くなる。

そこで、効用関数に一定の仮定をおいてこの誘因両立制約を簡素化することを考える。具体的には、SCP を満たす場合を考える。(ただし、パラメータ θ_i に関する順序づけが伊藤 1.3.2 と本稿との間で反

対になっているので、u_x が θ_i の厳密な*減少*関数であることがこの条件である。)[12]この数値問題では、サプライヤー i の効用 $U_i = w_i + u(x_i, \theta_i)$ を構成する関数 u について、$u(x_i, \theta_i) = -\theta_i x_i$ と特定化しているので、関数 u の x_i に関する一階微分 u_x は θ_i に対して厳密に*減少*関数となっている。したがって、実際、追加的な仮定なしにそのままで SCP を満たしているので、大域的条件の代わりに、局所的誘因両立制約を使って問題を簡単化することができる。

伊藤 1.3.2 では、この大域的条件を使ったセカンドベスト・モデルを、2 つの点で簡単化した上で問題を解いている。第 1 は伊藤 1.3.2 の問題(P_N)であり、LICD と LICU の両方の制約を置くものである。すなわち、表 2.6 における局所的条件(濃い灰色)、

$$U(\theta_i \mid \theta_i) \geq U(\theta_{i-1} \mid \theta_i) \qquad \forall i \tag{LICU}$$

$$U(\theta_i \mid \theta_i) \geq U(\theta_{i+1} \mid \theta_i) \qquad \forall i \tag{LICD}$$

だけを課すものである。[13]

第 2 は、伊藤 1.3.2 の問題(P'_N)であり、LICD と LICU のうちの 1 つの代わりとして、品質 x_i の解の単調性の条件(M_N)、

$$x_0 \geq \ldots \geq x_N$$

を置くものである。[14]LICD を例に取ると、それは各サプライヤーが効率性の点ですぐ 1 つ劣るタイプであると偽って申告するのを防ぐ条件だけを課すというものである。さらに手続き的には、単調性制約を無視して問題を解いた後に、得られた解が単調性を満たしていることを示すことにより、単調性制約(M_N)

---

[12] すなわち、$\forall x \in X$，$\forall \theta, \theta' \in \Theta$ に対して $\theta > \theta' \Rightarrow u_x(x,\theta) < u_x(x,\theta')$ が成り立つ場合。

[13] パラメータ θ に関する順序づけが伊藤 1.3.2 と本稿との間で反対になっているので、より効率的なタイプであると偽って申告させないための伊藤の条件 LICU 中に現れる $U(\theta_{i+1} \mid \theta_i)$ は、本稿では $U(\theta_{i-1} \mid \theta_i)$ に置き換えられているが本質的に同じものである。LICD についても同様である。

[14] パラメータ θ に関する順序づけが反対になっているので、不等号の向きが伊藤 1.3.2 の表記と逆になっているが本質的には何も違わない。

を無視することを正当化している。逆に単調性が満たされていないときは、LICD のほか、単調性制約か LICU かのいずれかを必ず含んでいなければならない。単調性については、伊藤 1.3.2 の仮定 1.4 に関連して、2.3.2 で詳しく説明する。

以下では実際に、前項で提示したファースト・ベスト・モデルを基にして、局所的条件 LICD を持つが、単調性(M_N)を外した形でセカンドベスト・モデルを解き、その後に解がこの単調性を満たしていることを確認する(`PS3_S.gms`)。

$$\max\ Util = \sum_i p_i\left[b(x_i) - w_i\right] \tag{2.1'}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$w_i - \theta_i x_i \geq ru \qquad \forall i \tag{2.3'}$$

$$w_i - \theta_i x_i \geq w_{i+1} - \theta_i x_{i+1} \qquad \forall i \tag{2.5}$$

なお、伊藤の問題(P'_N)では、参加制約(2.3')を最も非効率なサプライヤー(本稿の表記では $i = 2$)のみについて課しているが、本稿では前項のファースト・ベスト・モデルと同様にすべてのサプライヤーについてこれを課す(リスト 2.10 の 37 行目)。もちろん、伊藤が示すとおり、$i = 2$ 以外についてはこの制約式は無効であるから、解がゆがむおそれはない。一方、2.1.6 で触れたように、この制約式のスラック(左辺と右辺の差)を見ることで各サプライヤーが得る情報レントを確認することができる。LICD(2.5)は、入力ファイルの 38 行目に現れる。サプライヤー`i` から見て 1 つ効率性の点で劣る(添え字 `i` の順番では 1 つ後の)サプライヤーを表す添え字は、プログラム中では直感通り `i+1` とすればよい。

リスト 2.10: 3 タイプのセカンドベスト・モデル (`PS3_S.gms`)

```
...
 7 * Definition of Set
 8 Set    i      type of supplier       /0,1,2/;
 9 Alias (i,j);
10 * Definition of Parameters
11 Parameter
```

```
       Theta(i)       efficiency      /0      0.1
                                       1      0.2
                                       2      0.3/
       p(i)           probability of type
                                       /0      0.2
                                       1      0.5
                                       2      0.3/;
Scalar  ru            reservation utility    /0/;

* Definition of Primal/Dual Variables
Positive Variable
       x(i)          quality
       b(i)          maker's revenue
       w(i)          price;
Variable
       Util          maker's utility;
Equation
       obj           maker’s utility function
       rev(i)        maker's revenue function
       pc(i)         participation constraint
       licd(i)       incentive compatibility constraint;

* Specification of Equations
obj..  Util =e= sum(i, p(i)*(b(i)-w(i)));
rev(i)..b(i) =e= x(i)**(0.5);
pc(i).. w(i)-theta(i)*x(i) =g= ru;
licd(i)..w(i)-theta(i)*x(i) =g= w(i+1)-theta(i)*x(i+1);

* Setting Lower Bounds on Variables to Avoid Division by Zero
x.lo(i)=0.0001;;

* Defining and Solving the Model
Model SB3 /all/;
Solve SB3 maximizing Util using NLP;

* End of Model
```

この結果は、リスト 2.11 に示す出力ファイルのとおりである。

リスト 2.11: 3 タイプのセカンドベスト・モデルの出力ファイル(抜粋) (PS3_S.lst)

```
...
---- EQU rev  maker's revenue function

     LOWER     LEVEL     UPPER    MARGINAL
```

```

0     .       .       .      0.200
1     .       .       .      0.500
2     .       .       .      0.300

---- EQU pc  participation constraint

     LOWER     LEVEL     UPPER    MARGINAL

0     .       0.522     +INF       .
1     .       0.088     +INF       .
2     .        .       +INF     -1.000

---- EQU licd  incentive compatibility constraint

     LOWER     LEVEL     UPPER    MARGINAL

0     .        .       +INF     -0.200
1     .        .       +INF     -0.700
2     .        .       +INF       .

---- VAR x  quality

     LOWER     LEVEL     UPPER    MARGINAL

0 1.0000E-7    25.000     +INF  5.7724E-8
1 1.0000E-7     4.340     +INF      EPS
2 1.0000E-7     0.879     +INF      EPS

---- VAR b  maker's revenue

     LOWER     LEVEL     UPPER    MARGINAL

0     .       5.000     +INF       .
1     .       2.083     +INF       .
2     .       0.937     +INF       .

---- VAR w  price

     LOWER     LEVEL     UPPER    MARGINAL

0     .       3.022     +INF       .
1     .       0.956     +INF       .
2     .       0.264     +INF       .

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR Util          -INF      1.161     +INF       .
```

. . .	

2.2.4　ファーストベスト・モデルとセカンドベスト・モデルの解の比較

2 タイプ・モデルの結果について 2.1.6 で比較したときと同様に、ファーストベスト解と比較してセカンドベスト解がどのように変化したか、伊藤 1.3.2 の説明に照らしながら検討する(表 2.7)。なお、先にも述べたようにこのセカンドベスト・モデルは SCP を満たしているので、局所的制約モデルの解と後に述べる大域的制約モデルの解は等しい。

表 2.7:　3 タイプ・モデルのファーストベストとセカンドベストの比較

変数	プログラム中での変数名	サプライヤーのタイプ	セカンドベスト・モデルの解	ファーストベスト・モデルの解	両者の差
部品の品質	`x(i)`	0	25.00	25.00	0.00
		1	4.34	6.25	–1.91
		2	0.88	2.78	–1.90
メーカーの収入	`b(i)`	0	5.00	5.00	–0.00
		1	2.08	2.50	–0.42
		2	0.94	1.67	–0.73
メーカーの支払い価格	`w(i)`	0	3.02	2.50	0.52
		1	0.96	1.25	–0.29
		2	0.26	0.83	–0.57
メーカーの効用	`Util`		1.16	1.37	–0.21
サプライヤーの効用(=情報レント)	`pc.l(i) -pc.lo(i)`	0	0.52	0.00	0.52
		1	0.09	0.00	0.09
		2	0.00	0.00	0.00
誘因両立制約のラグランジュ乗数	`licd.m(i)`	0	–0.20	–	–
		1	–0.70	–	–
		2	0.00	–	–
参加制約のラグランジュ乗数	`pc.m(i)`	0	0.00	0.00	–
		1	0.00	0.00	–
		2	–1.00	–1.00	–

セカンドベスト解をファーストベスト解と比べると、品質 `x(i)`は、最も効率的なサプライヤー(タイプ 0)についてのみ一定で、そのほかの非効率なタイプにおいて低下している。これは、メーカーの支払い価格 `w(i)`がタイプ 0 についてのみ引き上げられ、他は引き下げられたことと対応する。

このように、効率的なサプライヤーが、より非効率的なサプライヤーの振りをすることを阻止するために、メーカーは彼らの行動を予測して以前よりも気前のよいオファーを出さざるを得なくなる。ただし、このメーカーのオファーは、サプライヤーが自らのタイプを偽っても偽らなくても無差別な水準に決定することが最適である(過剰に気前よくなる必要はない)ので、誘因両立制約 `licd(i)`はタイプ 0, 1 についてのみ有効になる(すなわちそれらのラグランジュ乗数 `licd.m(i)`はゼロではない)。最も非効率的なサプライヤーは(偽るべき、より非効率的なサプライヤーが存在しない以上)、以前と同様に留保効用水準だけを獲得できるオファーしか得られない。このことは、参加制約のラグランジュ乗数の値がタイプ 2 についてのみ有効(ラグランジュ乗数がゼロではない)であることから確認できる。

より効率的なサプライヤー(ここではタイプ 0, 1)が獲得する情報レントは、`EQU` ブロックの参加制約式 `pc(i)`の `LEVEL` と `LOWER` の差として示される(リスト 2.11 の 137–138 行目)。また、タイプ 0, 1 については誘因両立制約が等号で成立し、かつ、留保効用 `ru` を 0 としているので、これらのタイプの効用はこの情報レントに等しい。(最も非効率なタイプ 2 の効用は、情報レントさえも得られないので留保効用のまま、すなわち 0 である。)タイプ 1 は、セカンドベストでは価格が低下しているにも関わらず、正の効用を得ている。これは、セカンドベストではタイプに関する情報の非対称性を利用して、より非効率的なサプライヤー(タイプ 2)の振りをすることで、ファーストベストのときにはゼロであった情報レントを得ることにより、効用が正に転じたものである。最も効率的なサプライヤーについて効用が増加していることは言うまでもない。メーカーの効用は、最終的にこうした情報の非対称性に起因する非効率性と、情報レントを通じた収奪を受けることによって、ファーストベストの時よりも低い水準となる。

ところで、伊藤(2003)では品質 x_i に課されていた単調性の制約(M_N)をいったん無視して問題を解き、その後で解が単調性を満たしていることを確認していた。表 2.7 のセカンドベスト解を見てみると、実際、効率的なサプライヤーほどより高い品質を提供していることが確認できる。

2.2.5 大域的誘因両立制約を用いたセカンドベスト・モデル (`PS3_S_GIC.gms`)

本項では、前項のモデルの中の、局所的誘因両立制約を大域的誘因両立制約に置き換えたモデル、すなわち大域的誘因両立制約モデルを解き、両モデルの解の比較を行う。リスト 2.10 の入力ファイル

の理解が十分であれば、新しいモデル(リスト 2.12)の変更点だけを理解しておけばよい。具体的には、38 行目が(2.4') に対応したものに書き換えられている。ただし、ここで制約式名 `ic` の添え字が `i` と `j` の 2 つになっていて、それらはコンマ「`,`」で区切られていることに注意する。プログラム中の制約式の内容と(2.5)を比較してわかるように、`ic(i,j)` の意味は、タイプ `i` のサプライヤーがタイプ `j` のサプライヤーの振りをする(ただし、自己のタイプの場合も含む: `i=j`)ことを阻止するための制約である。

リスト 2.12: 3 タイプのセカンドベスト・モデル—大域的条件の場合(抜粋) (`PS3_S_GIC.gms`)

```
...
28 Equation
29        obj           maker’s utility function
30        rev(i)        maker's revenue function
31        pc(i)         participation constraint
32        ic(i,j)       incentive compatibility constraint;
33
34 * Specification of Equations
35 obj..   Util =e= sum(i, p(i)*(b(i)-w(i)));
36 rev(i)..b(i) =e= x(i)**(0.5);
37 pc(i).. w(i)-theta(i)*x(i) =g= ru;
38 ic(i,j)..w(i)-theta(i)*x(i) =g= w(j)-theta(i)*x(j);
39
40 * Setting Lower Bounds on Variables to Avoid Division by Zero
...
```

大域的条件を課した場合でも、SCP が満たされているモデルでは、局所的条件を課したモデルの解と等しくなる。ただ、制約式が `licd(i)`から `ic(i,j)`に変更されているので、そのラグランジュ乗数の表示が若干変わる(リスト 2.13)。6 つの誘因両立制約のうちで有効なものは、タイプ 0 がタイプ 1 を偽るもの(145 行目)と、タイプ 1 がタイプ 2 を偽るもの(148 行目)の 2 本である。表現が異なるものの、これらのラグランジュ乗数の値が表 2.7 に示したものと同じであることがわかる。さらに付言すれば、自分よりもひとつだけ効率性の点で劣るタイプを偽る誘因両立制約のみが有効である。これは、そのほかの誘因両立制約を課さずにこの問題を解いたとしても(=局所的条件 LICD(2.5)のみであったとしても)、同じ品質や価格といった解を得ることができることを意味する。すなわち、2 つのプログラム(リスト 2.10, 2.11)が同じ解を導き出すことが、この点でも確認できる。

リスト 2.13: 3 タイプのセカンドベスト・モデルの解—大域的条件の場合(抜粋) (PS3_S_GIC.lst)

```
...
---- EQU pc  participation constraint

     LOWER     LEVEL     UPPER    MARGINAL

0      .       0.522     +INF       .
1      .       0.088     +INF       .
2      .         .       +INF     -1.000

---- EQU ic  incentive compatibility constraint

       LOWER     LEVEL     UPPER    MARGINAL

0.1      .         .       +INF     -0.200
0.2      .       0.346     +INF       .
1.0      .       2.066     +INF      EPS
1.2      .         .       +INF     -0.700
2.0      .       4.478     +INF       .
2.1      .       0.346     +INF       .
...
```

2.3　モデルの拡張

これまで伊藤のモデルに従ってモデルを構築してきた。とくに、伊藤では解析的に分析をすることを前提にして、モデルを可能な限り簡単にするようにいくつかの仮定がおかれていた。その第 1 は、Spence-Mirrlees の単一交差性(SCP)であり、これによって考慮すべき制約式を大幅に少なくしていた。第 2 は、品質 x_i に対して課された単調性制約をいったん無視して問題を解き、事後的に解がこの単調性を満たしていること確認するという手法を取っていた。より厳密に言うと、単調性が成り立つための条件を仮定していた(伊藤 1.3.2)。ここでは、これらの条件が満足されない状況を前提として、その際にどのようにモデルを解くべきかを考える。

本節では、以上の 2 つの簡単化の仮定に関して、2.3.1 では SCP が成立しないモデルを扱い、2.3.2 では SCP が成立しているモデルにおける単調性の問題を議論する。加えて、2.3.3 ではタイプの数が 3 つ以上の場合にどのようにモデルを構築するかについて触れる。

2.3.1 セカンドベスト・モデル：大域的誘因両立制約を用いた場合（PS3_S_SPC.gms）

ここでは SCP を満たさない場合を考えてみる。モデルの構造自体は、2.2.5 で提示した大域的誘因両立モデルと同一である。ただし SCP を満たさない状況を考えるために、サプライヤー i の効用関数 $U_i = w_i + u(x_i, \theta_i)$ において、関数 u を $u(x_i, \theta_i) = -\theta_i x_i$ と特定化してきたものを、

$$u(x_i, \theta_i) = -\left[\theta_i + \left(1 - \theta_i + \theta_i^2\right)x_i\right]$$

に置き換える。(ただし、$0 < x_i < 1$, $0 < \theta_i < 1$ とする。)この関数形を用いることで、u の x_i および θ_i に関する交差偏微分の値 $u_{x\theta} = 1 - 2\theta_i$ が、$\theta_i = 0.5$ より小さければ正、大きければ負と符号が変わるので、伊藤の仮定 1.2 は満足されずに SCP は成立しない。ただし、伊藤の仮定 1.3(u が θ の厳密な*減少*関数であること)は満足する。[15]このときのメーカーの効用最大化問題は、

$$\max\ Util = \sum_i p_i \left[b(x_i) - w_i\right] \tag{2.1'}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$w_i - \left[\theta_i + \left(1 - \theta_i + \theta_i^2\right)x_i\right] \geq ru \qquad \forall i \tag{2.3''}$$

$$w_i - \left[\theta_i + \left(1 - \theta_i + \theta_i^2\right)x_i\right] \geq w_j - \left[\theta_i + \left(1 - \theta_i + \theta_i^2\right)x_j\right] \qquad \forall i \neq j \tag{2.4'}$$

となる。

プログラム(リスト 2.14, `PS3_S_SCP.gms`)中では、参加制約(2.3”)の名前は従来通り `pc(i)`、大域的誘因両立制約(2.4’)は `ic(i,j)`となっている。(プログラム中の `sqr(…)`は自乗の関数である。)52 行目で、この(2,1’), (2.2), (2.3”), (2.4’)からなる大域的誘因両立制約モデルの名前を `SB_gic_wo_SCP` として定義している。

比較のために、本来課すべき大域的条件(2.4’)の代わりに局所的条件(LICD, LICU)、

---

[15] パラメータ に関する順序づけが伊藤 1.3.2 と本稿との間で反対になっているので、満たすべき条件が「増加関数」ではなく、「減少関数」と反対になっているが、本質的には同じである。

$$w_i - \left[\theta_i + \left(1-\theta_i+\theta_i^2\right)x_i\right] \geq w_{i+1} - \left[\theta_i + \left(1-\theta_i+\theta_i^2\right)x_{i+1}\right] \quad \forall i$$

$$w_i - \left[\theta_i + \left(1-\theta_i+\theta_i^2\right)x_i\right] \geq w_{i-1} - \left[\theta_i + \left(1-\theta_i+\theta_i^2\right)x_{i-1}\right] \quad \forall i$$

を課したモデルも構築している。これら 2 つの局所的条件は、`licd(i)`, `licu(i)`として 43–46 行目で示されている。53 行目では `ic(i,j)`の代わりに `licd(i)`, `licu(i)`を用いた局所的誘因両立制約モデルの名前を `SB_lic_wo_SCP` として定義している。

リスト 2.14: SCP を満たさない場合のモデル(抜粋) (`PS3_S_SCP.gms`)

```
...
10 | * Definition of Parameters
11 | Parameter
12 |        theta(i)       efficiency      /0      0.1
13 |                                       1       0.4
14 |                                       2       0.9/
15 |        p(i)          probability of type
16 |                                       /0      0.2
17 |                                       1       0.5
18 |                                       2       0.3/;
...
28 | Equation
...
32 |        ic(i,j)        incentive compatibility constraint
33 |        licd(i)        incentive compatibility constraint
34 |        licu(i)        incentive compatibility constraint;
35 |
36 | * Specification of Equations
37 | obj..   Util =e= sum(i, p(i)*(b(i)-w(i)));
38 | rev(i)..b(i) =e= x(i)**(0.5);
39 | pc(i)..  w(i)  -(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i))
40 |            =g= ru;
41 | ic(i,j)..w(i)  -(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i))
42 |     =g=  w(j)  -(theta(i)+(1-theta(i)+sqr(theta(i)))*x(j));
43 | licd(i)..w(i)  -(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i))
44 |     =g=  w(i+1)-(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i+1));
45 | licu(i)..w(i)  -(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i))
46 |     =g=  w(i-1)-(theta(i)+(1-theta(i)+sqr(theta(i)))*x(i-1));
47 |
48 | * Setting Lower Bounds on Variables to Avoid Division by Zero
49 | x.lo(i)=0.0001;
50 |
51 | * Defining and Solving the Model
```

```
Model SB_gic_wo_SCP /obj, rev, pc, ic/;
Model SB_lic_wo_SCP /obj, rev, pc, licd, licu/;

Solve SB_gic_wo_SCP maximizing Util using NLP;
Solve SB_lic_wo_SCP maximizing Util using NLP;
* End of Model
```

大域的制約モデルを解いて得られた x_i の解は単調性を満たしていない(表 2.8)が、そもそも局所的条件を用いているわけではないので単調性を満たしている必要はなく、得られた解がそのまま正しい解である。大域的制約モデルと局所的制約モデルの違いを誘因両立制約の点からみていこう。大域的制約モデルでは、3 つの場合(タイプ 0 が「タイプ 2」と表明、タイプ 1 が「タイプ 0」と表明、および、タイプ 1 が「タイプ 2」と表明)に、誘因両立制約のラグランジュ乗数がゼロでないので、自らのタイプを偽る誘因が存在することがわかる。一方、局所的制約モデルでは、この 3 つの場合のうちタイプ 0 が「タイプ 2」と表明する場合を防ぐ制約が欠落している。(そのほか、タイプ 2 がタイプ 0 を偽ることを防ぐ制約も欠落しているが、この数値例の結果ではこの制約は有効でないため、この制約式が欠落していることで解がゆがむことはない。)以上の結果、局所的制約モデルの解は、大域的制約モデルの解から乖離して誤ったものとなる。局所的制約モデルが誤ったモデルである以上この解自体に意味はないが、必要な制約式が欠落しているために目的関数の値 `Util` はより大きくなっている。

表 2.8: 単一交差性を満たさない場合のモデル間の解の比較

変数	GAMS プログラム中での変数名	サプライヤーのタイプ	大域的制約モデル	局所的制約モデル
部品の品質	`x(i)`	0	0.22	0.30
		1	0.43	0.51
		2	0.22	0.16
メーカーの収入	`b(i)`	0	0.47	0.55
		1	0.66	0.71
		2	0.47	0.40
メーカーの支払い価格	`w(i)`	0	1.10	1.12
		1	1.26	1.31
		2	1.10	1.04
メーカーの効用	`Util`		–0.62	–0.61
サプライヤーの効用(=情報レント)	`pc.l(i) -pc.lo(i)`	0	0.80	0.75
		1	0.53	0.52
		2	0.00	0.00
大域的誘因両立制約のラグランジュ乗数	`ic.m(i,j)`	0 が 1 と表明	0.00	–
		0 が 2 と表明	–0.40	–
		1 が 0 と表明	–0.20	–
		1 が 2 と表明	–0.30	–
		2 が 0 と表明	0.00	–
		2 が 1 と表明	0.00	–
局所的誘因両立制約のラグランジュ乗数	`licd.m(i)`	0	–	–0.20
		1	–	–0.70
		2	–	0.00
	`licu.m(i)`	0	–	0.00
		1	–	0.00
		2	–	0.00
参加制約のラグランジュ乗数	`pc.m(i)`	0	0.00	0.00
		1	0.00	0.00
		2	–1.00	–1.00

2.3.2 単調性が成り立つための十分条件を仮定できないとき（`PS3_S_MN.gms`）

伊藤 1.3 では、SCP を満たすモデルにおいて、品質 x_i について単調性が成り立つような十分条件

(伊藤 1.3.2 の仮定 1.4)をおいて問題を簡単化して解くことを考えていた。[16]しかし、実際の問題を考えたときにこの十分条件が成り立つとは限らない。[17](その一方で、ここで考えていることは単調性が成り立つための十分条件であるから、この十分条件を満たさない場合であっても単調性が成り立つ場合もあり得る。)このように単調性が成り立つことを仮定できない場合には、明示的に単調性の条件(M_N)(2.6)(あるいは、LICU: $w_i - \theta_i x_i \geq w_{i-1} - \theta_i x_{i-1}$)を課してモデルを解くことになる。(ここでは、伊藤に倣い、SCP の成立する場合を考える。)

$$\max\ Util = \sum_i p_i \left[b(x_i) - w_i \right] \tag{2.1'}$$

subject to

$$b(x_i) = x_i^{0.5} \qquad \forall i \tag{2.2}$$

$$w_i - \theta_i x_i \geq ru \qquad \forall i \tag{2.3'}$$

$$w_i - \theta_i x_i \geq w_{i+1} - \theta_i x_{i+1} \qquad \forall i \tag{2.5}$$

$$x_i \geq x_{i+1} \qquad \forall i \tag{2.6}$$

解析的に解く場合には問題が複雑化するが、数値計算による場合には簡単である。実際、この(2.6)に相当する制約式 `mn(i)`を、元のプログラム(`PS3_S.gms`)に追加して解けばよい(`PS3_S_MN.gms`):

```
mn(i)..        x(i) =g= x(i+1);
```

---

[16] この仮定は、(伊藤の θ に関する順序づけをそのまま用いて)(i)関数 $\Phi(\cdot)$ が準凸であることと、(ii)関数 $\Phi(\cdot)$ が (x, θ) に関して、すべての i について差分増加であることである。ここで、

$$\Phi(x_i, \theta_i) = S(x_i, \theta_i) - \left[(1 - F_i)/p_i \right] \left[u(x_i, \theta_{i+1}) - u(x_i, \theta_i) \right],$$

$$F_i = 1 - \sum_{j=i+1}^{N} p_j = \sum_{j=0}^{i} p_j\,,$$

$$F_{-1} = 0$$

である。

[17] この条件を満たす可能性は、(事前確率の設定の仕方がまったくのランダムであるとするならば)タイプ数が多くなるほど急速に小さくなる。この可能性がどの程度のものであるかについては補論で例示する。

ここで、元のプログラム中のパラメータ設定をそのまま用いた場合には、もともと単調性を満たすことになるので、リスト 2.10 に `mn(i)`を追加して解いた場合にも解は以前とまったく同じで、制約式 `mn(i)`は一切有効にはならないことに注意しておく。

しかし、この結果は種々のパラメータ設定に依存したものであることに注意する必要がある。たとえば、事前確率 p_i のみを、

$$\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 0.30 \\ 0.10 \\ 0.60 \end{pmatrix}$$

と変えたとき、あるいは、効率性のパラメータ θ_i のみを、

$$\begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 0.10 \\ 0.30 \\ 0.31 \end{pmatrix}$$

と変えたときには、(LICD に加えて、単調性または LICU の制約を課すことなしには)単調性が成立しないので正しい解は得られない。

2.3.3 タイプ数が多いとき (`PS10_S.gms`)

タイプ数が多くなると、追加的仮定無しに解析的に解くことは困難になる一方、数値計算においては、(実際のパラメータ θ_i や事前確率 p_i がどのようなものであるべきかという問題を除けば)プログラムの修正は最小限ですむ。たとえば、10 タイプ・モデル($i = 0,1,2,...,9$)の場合には、タイプを表す添え字の集合を、タイプ数に応じて、

```
Set    i      type of supplier      /0*9/;
```

と書き換えればよい(リスト 2.15)。ここで「*」(アスタリスク)は連続した数の添え字の入力を簡略化するための記号である。上のようなプログラムの代わりに、

```
Set    i      type of supplier      /0,1,2,3,4,5,6,7,8,9/;
```

としても同じである。また、100 タイプ・モデルを解く場合には、最後の「`9`」の代わりに「`99`」とすればよい(ただし、試用版の GAMS を用いた場合には、モデルの大きさの制約のために 100 タイプ・モデルは解くことができない)。この入力ファイル(`PS10_S.gms`)の 14–15 行目では、効率性のパラメータ θ_i や事前確率 p_i について、タイプ数がいくつになった場合でも数値計算上困難にならない値(一定の率で逓増する θ_i と均一の p_i)を自動で計算するようになっているが、これらに代えて個別に値を与えてもよい。[18]

リスト 2.15: 10 タイプ・モデルの入力ファイル(抜粋) (`PS10_S.gms`)

```
...
   7 * Definition of Set
   8 Set     i       type of supplier        /0*9/;
   9 Alias (i,j);
  10 * Definition of Parameters
  11 Parameter
  12        theta(i)       efficiency
  13        p(i)           probability of type;
  14 theta(i)=ord(i)/card(i);
  15 p(i)=1/card(i);
...
```

最後に、これまでに構築してきたモデルと入力ファイルの一覧を表 2.9 に示す。これらのサンプル・プログラムは付録として本稿とは別に提供されている。

表 2.9: 部品調達問題に関するモデル一覧

入力ファイル名	タイプ数	非対称情報	誘因両立制約	単調性の制約	伊藤	本稿	備考
PS2_F_eff.gms	2	なし	-	-	1.1.2	2.1.2	効率的サプライヤー
PS2_F_inf.gms	2	なし	-	-	1.1.2	2.1.3	非効率的サプライヤー
PS2_F.gms	2	なし	-	-	1.1.2	2.1.4	

[18] プログラム中の `ord(i)`と `card(i)`は、それぞれ、添え字 `i` がその集合の中で現れる順番、および、その集合内の要素の数を表す GAMS の関数である。

PS2_S.gms	2	あり	局所的	なし	1.1.4	2.1.5	
PS2_F&S.gms	2	なし/あり	局所的	なし	1.1.2/1.1.4	2.1.7	
PS3_F.gms	3	なし	-	-	-	2.2.2	
PS3_S.gms	3	あり	局所的	なし	1.3.2	2.2.3	
PS3_S_GIC.gms	3	あり	大域的	-	-	2.2.5	
PS3_S_SCP.gms	3	あり	大域的/局所的	-	-	2.3.1	局所的制約モデルの解は正しくない
PS3_S_MN.gms	3	あり	局所的	あり	-	2.3.2	
PS10_S.gms	10	あり	局所的	なし	-	2.3.3	
PS10_S_MN.gms	10	あり	局所的/大域的	あり/なし	-	補論	単調性制約なしのモデルの解は正しくない

3. むすび―結論に代えて

本稿では、アドバース・セレクションの基本的なモデルについて、タイプが 2 個の数値モデルと 3 個の数値モデルを作って解いてきた。仮定された関数形とパラメータに限定されるとはいえ、その下での計算結果は、伊藤の理論モデルの解析的説明と一致している。すなわち、ファーストベストに比べて、セカンドベストでは最も効率的なタイプのみが品質を維持する。また、彼らは、より非効率的なタイプを偽ることが可能であるために、そこから発生する情報レントを獲得する。一方、最も非効率的なタイプは情報レントを得ることができない。この結果、メーカーの効用は低下する。

本稿で、われわれは、モデルを非線形計画問題として定式化して数値計算を行った。こうしたプログラミング・モデル・アプローチは、いくつかの点で有用であることがわかる。第 1 に、セカンドベストの状況における最も重要な要素は、上で触れたように、情報の非対称性に起因する情報レントである。この情報レントは、GAMS の出力ファイル中で、制約式に関する計算結果を示す `EQU` ブロックにある、`LEVEL` 値と `LOWER` 値の差、すなわち、スラックとして陽表的に示される。第 2 に、タイプの数が増加すると誘因両立制約が複雑になる。伊藤(2003)では、効用関数に関する SCP の仮定を導入することにより、数多い大域的条件を、より少数の局所的条件に置き換えている。コンピュータ・プログラム中では、こうした追加的な仮定の有無を簡単にプログラムに反映させることができ、それによって、確かに大域的制約モデルと局所的制約モデルの間で解が一致すること、あるいは逆に、これらの追加的な仮定が成立しないときには解が一致しないことを示すことができる。

第 3 に、われわれのアプローチでは、解析的方法において用いられる簡単化のための工夫の必要性

は、それほど高くない。すなわち、これらの理論分析でよく用いられるアプローチを離れた場合に元の大きなモデルをそのまま解いたとしても、追加的費用は気にするほどのものではない。タイプ数をさらに多く(たとえば 10 個に)したとき、理論的分析は急速に難しくなるが、数値計算上の問題は「追加的に何秒の時間がかかるか」ということでしかない。第 4 に、数値例による分析に限界があることはいうまでもないが、しかしながら、補論において行う単調性が満たされる可能性を推量するためのモンテカルロ・シミュレーションのような手法は、数値計算モデルにしてはじめて行えることである。

補論: 単調性が成り立つための十分条件について

本文 2.3.2 において単調性が成り立たない場合について考察した。ここでは、単調性に関して以下の 2 点について追加的に検討する。第 1 点は、単調性が成り立つための十分条件を導き出すことである。第 2 点は、具体的にどの程度の確率で単調性が成り立つのかを示すことである。後者については、2 つの段階で検討する。第 1 は、(A)具体的に*どの程度の確率で単調性の十分条件が成り立つか*である。第 2 は、(B)単調性の*十分条件の成立・不成立に関わらず、どの程度の確率で解が単調性を満たすか*である。ただし、伊藤 1.3 におけるタイプ θ_i の順序づけと、本稿におけるそれが異なっているために、検討すべき命題そのものについて予備的な考察を A.1 で行う。続いて、A.2 で具体的な確率について検討する。

A.1 (タイプに関する順序づけを変更した場合の)単調性が成り立つ十分条件

第 1 点の単調性が成り立つための十分条件(伊藤 1.3.2 および 1.4)を、タイプ θ_i に関する順序づけが反対になっている場合について検討する。伊藤における θ_i をここでは $\alpha_i\left(=-\theta_{N-i}<0\right)$、そのタイプの出現確率を $q_i\left(=p_{N-i}\right)$、品質を $y_i\left(=x_i\right)$ と表記することにする。α_i は大きいほど効率的という順序づけになる。限界費用は、$\theta_0<\ldots<\theta_N$ とすると、α_i を用いたときの表現は、$\alpha_0=-\theta_N<\alpha_i=-\theta_{N-1}<\ldots<\alpha_N=-\theta_0$ となる。これによって、伊藤 1.3 で定義されている関数 Φ は、

$$\begin{aligned}\Phi\left(y_i,\alpha_i\right)&=S\left(y_i,\alpha_i\right)-\frac{1-G_i}{q_i}\left[u\left(y_i,\alpha_{i+1}\right)-u\left(y_i,\alpha_i\right)\right]\\&=b\left(y_i\right)+\alpha_i y_i-\frac{1-G_i}{q_i}\left(\alpha_{i+1}-\alpha_i\right)y_i\end{aligned}\qquad\text{(A.1)}$$

ただし、$G_i=\Pr\left\{\alpha\le\alpha_i\right\}=\sum_{k=N-i}^{N}q_k=1-F_{N-i-1}$, $G_N=1$

と表現し直すことができ、これが y_i と α_i に関して差分増加であること(すなわち、$\alpha_{i+1}>\alpha_i$ ならば、任

意の y_i に対して $\Phi_y(y_i,\alpha_{i+1})-\Phi_y(y_i,\alpha_i)\geq 0$)が単調性を満たす条件であった。(A.1)を y_i について微分すると、タイプ i について、

$$\Phi_y(y_i,\alpha_i)=b'(y_i)+\alpha_i-\frac{1-G_i}{q_i}(\alpha_{i+1}-\alpha_i) \tag{A.2}$$

同様に、タイプ $i+1$ について、

$$\Phi_y(y_i,\alpha_{i+1})=b'(y_i)+\alpha_{i+1}-\frac{1-G_{i+1}}{q_{i+1}}(\alpha_{i+2}-\alpha_{i+1}) \tag{A.3}$$

(A.3)から(A.2)の両辺をそれぞれ差し引くと、

$$\Phi_y(y_i,\alpha_{i+1})-\Phi_y(y_i,\alpha_i)=(\alpha_{i+1}-\alpha_i)-\frac{1-G_{i+1}}{q_{i+1}}(\alpha_{i+2}-\alpha_{i+1})+\frac{1-G_i}{q_i}(\alpha_{i+1}-\alpha_i) \tag{A.4}$$

すべての i について α_i の区間幅が同じ($(\alpha_{i+1}-\alpha_i)=(\alpha_{i+2}-\alpha_{i+1})=(\theta_{i+1}-\theta_i)=\Delta\alpha>0$)である状況に限定した場合を考えると、(A.4)の 3 つの括弧内が等しくなるので、伊藤 1.4.2 の単調危険率条件(monotone hazard rate condition, MHRC)の離散形版に相当する条件、

$$\frac{1-G_i}{q_i}\geq\frac{1-G_{i+1}}{q_{i+1}} \tag{A.5}$$

が成立すれば、$\Phi_y(y_i,\alpha_{i+1})-\Phi_y(y_i,\alpha_i)\geq 0$ が言える。

ところで、元の θ_i の分布関数 F_i は、

$$F_i=\Pr\{\theta\leq\theta_i\}=\sum_{j=0}^{i}p_j$$

$$F_N=1$$

であり、たとえば、

θ_0 については、$F_0=p_0$

θ_1 については、$F_1=p_0+p_1$

…

であった。一方、α_i の分布関数 G_i は、

$\alpha_0(=-\theta_N)$ については、$G_0 = q_0 = p_N = 1 - F_{N-1}$

$\alpha_1(=-\theta_{N-1})$ については、$G_1 = q_0 + q_1 = p_N + p_{N-1} = 1 - F_{N-2}$

…

であるから、$G_i = 1 - F_{N-i-1}$ となる。したがって、危険率の条件(A.5)は、

$$\frac{F_{N-i-1}}{p_{N-i}} \geq \frac{F_{N-i-2}}{p_{N-i-1}} \tag{A.6}$$

と書き換えることができる。これが成立することが単調性が成り立つための十分条件となる。

A.2 単調性が成り立つ確率

以上をふまえて、具体的にどの程度の確率で単調性が成り立つのかを検討するために 10 タイプで θ_i の区間幅が一定であるモデル(`PS10_S_MN.gms`)を作る。第 1 段階の問題は、どの程度の確率で、単調性が成り立つための十分条件(A.6)が満たされているかを調べるものである。そのため、各タイプ i の事前確率 p_i について(一般には θ_i 等の設定と同様に実証的問題であるが)、ここでは区間[0,1]上の一様分布に従うものとして t=1,000 回のモンテカルロ・シミュレーションを行う。第 2 段階の問題を考えるため、単調性の制約 `mn(i)` を課さないモデル(`SB_lic`)と課したモデル(`SB_lic2`)の 2 つを作り、それらの解を比較する。(なお、後者は常に正しい解を導く。)両者の解が一致していることは、(単調性の十分条件の成立・不成立に関わらず)単調性が成立していることを意味している。したがって、第 2 段階の問題は、どの程度の確率で 2 つのモデルの解が一致するかということになる。なお、この問題を考察するモデルの入力ファイル(`PS10_S_MN.gms`)が込み入っているのは、モンテカルロ・シミュレーションのために事前確率 `p(i)` をランダムに決めるためのプログラムが必要であることと、単調性の制約 `mn(i)` を課したものと課さないものの 2 つを、1 つのモデルで解いているからである。

第 1 段階の問題(A)まったくランダムに事前確率 p_i を設定したときどの程度の確率で MHRC(A.6) が成立するかについては、この数値例では、1,000 回の試行を行ったときに、MHRC を満足する場合

はただの 1 回(0.1%)である。[19]ただし、MHRC の成立はあくまでも十分条件であるから、これを満たさなくても単調性が成立する、すなわち、上の 2 つのモデルの解が一致するかも知れない。そこで、第 2 段階の問題として、(B)2 つのモデルの解が一致するのはどの程度の頻度であるかをみる。この数値例では、(B)が発生する場合は 1,000 回の試行で 7 回(0.7%)である。もちろん、この計算は一連の仮定に基づいているものである。たとえば、タイプ数を減ら(増や)すと、これらの確率が増加(減少)する。同じ設定で 5 タイプのモデルを考える(プログラム中では、8 行目の「`Set i type of supplier /0*9/`」の末尾を「`/0*4/`」と書き換えればよい)と、(A)が 24%、(B)が 36%程度になる。結論の一般性については十分に留保する必要があることは承知の上であるが、MHRC(A.6)を仮定して解くことができる場合や、単調性を無視して解くことができる場合は、意外と限られたものであると推量される。

[19] なお、MHRC(A.6)は θ_i と p_i の関係だけについての条件であるから、その条件が満足されるかどうかは最大化問題の内容(たとえば、効用関数の関数型の選択)とは無関係に決まることである。

謝辞

本研究は、橋本が政策研究大学院大学において客員研究員として行った研究成果を含む。一連の研究支援に対して、ここに記して謝意を表す。

参考文献

細江宣裕，我澤賢之，橋本日出男（2004）『テキストブック応用一般均衡モデリング―プログラムからシミュレーションまで―』，東京大学出版会.

伊藤秀史（2003）『契約の経済理論』，有斐閣.

Brooke, A., D. Kendrick, A. Meeraus, R. Raman, R. E. Rosenthal (2010) *GAMS A user's Guide*, GAMS Development Corporation.

McCarl, B. A. (2009) *McCarl Expanded GAMS User Guide*, version 23.3. <http://www.gams.com/dd/docs/bigdocs/gams2002/mccarlgamsuserguide.pdf> (2011 年 1 月 8 日取得)

Varian, H. (1992) *Microeconomic Analysis*, 3rd Ed., Norton.

Contract Theory: A Programming–Model Approach

February 24, 2011

Hideo Hashimoto[a], Kojun Hamada[b], Nobuhiro Hosoe[c]

Abstract

This is a study to develop and solve numerical models based on Itoh's (2003, Ch. 1) "Parts Supply Problems" for better understanding the contract theory. In the first part of this paper, by following Itoh (2003) we investigate 2- and 3-agent type cases; in the succeeding part, by using numerical examples, we examine how likely the simplifying assumptions often used in theoretical analysis are to hold. Finally, we demonstrate that we can extend these basic models to ones with a much larger number of agent types easily by exploiting the merit of our programming–model approach.

Keywords:

Principal-agent problem, adverse selection, numerical model, single-crossing property, monotonicity

[a] Professor Emeritus, Osaka University.

[b] Associate Professor, Faculty of Economics, Niigata University.

[c] Associate Professor, National Graduate Institute for Policy Studies. Author correspondence: 7-22-1 Roppongi Minato, Tokyo 106-8677, Japan. E-mail: nhosoe@grips.ac.jp.