GAMS における CGE モデルのデバッグ方法

武田史郎
京都産業大学経済学部
2013/06/25

1. はじめに

デバッグ（debug）とはプログラムのバグ（bug、欠陥、誤り）を修正する作業のことである。一口にデバッグといっても、プログラムの文法の誤り（シンタックス・エラー）を修正する、自分の意図通りのモデルになるようにチェックをする、データのチェックをするなど、様々なケースがありうる。このうち、シンタックス・エラーについては GAMS を実行する際にエラーメッセージが表示され、そのメッセージに従って誤りを修正すればよいので、比較的対処しやすい（シンタックス・エラーについては、GAMS User's Guide や McCarl GAMS User Guide が詳しい）。これに対して、モデルやデータ自体の誤り、すなわち経済学的に意味のあるモデル、データになっているのか、自らの意図するモデル、データになっているのかについては、必ずしもエラーメッセージが出るわけではないので、デバッグするのが比較的難しい。本稿では、後者の意味でのデバッグの方法、特に benchmark replication によるデバッグ方法について説明する。

2. サンプルのコード

本稿では表 1 のコードを例にとり説明する[1]。MCP で記述したモデルも NLP で記述したモデルもどちらもモデル自体は同じである。

表 1：サンプルのコード

プログラムのファイル名	説明
how-to-debug-mcp.gms	MCP で記述したモデル
how-to-debug-corrected-mcp.gms	バグ修正済みの MCP で記述したモデル
how-to-debug-nlp.gms	NLP で記述したモデル
how-to-debug-corrected-nlp.gms	バグ修正済みの NLP で記述したモデル

3. モデルの説明

[1] コードは次の場所からダウンロードできる。
http://shirotakeda.org/assets/files/research/note/HowToDebug_20130105.zip

以下、サンプルのプログラムにおけるモデルを説明する。モデルの特徴は以下の通りである。

- 完全競争、閉鎖経済モデル。
- 3 部門・3 財モデル。
  - AGR, MAN, SER の 3 つの部門があり、それぞれが一種類の財を生産している。
- 各部門は資本と労働を用いて、CES 生産関数に従い生産をおこなう。
- 一つの代表的家計が存在する。
  - 家計は CES 効用関数を持ち、3 つの財を消費する。
  - 家計は生産要素の提供により所得を得る。
- 一つの政府が存在する
  - 政府は政府支出をおこなう
  - 生産税、消費税、一括税により税を徴収している。

プログラム内では、モデルは、MCP 形式（MCP ソルバーで解く形式）、NLP 形式（NLP ソルバーで解く形式）の二つが定義されている（それぞれ model_mcp と model_nlp という名前）。

3.1. ベンチマーク・データ

ベンチマーク・データは表 2 で与えられる。

表 2：ベンチマーク・データ

	agr	man	ser	u	g	hh	gov	Sum
agr	200	-10	-20	-150	-20	0	0	0
man	-20	300	-50	-180	-50	0	0	0
ser	-30	-80	500	-350	-40	0	0	0
u	0	0	0	700	0	-700	0	0
g	0	0	0	0	110	0	-110	0
k	-70	-120	-100	0	0	290	0	0
l	-70	-70	-300	0	0	440	0	0
ty	-10	-20	-30	0	0	0	60	0
tc	0	0	0	-20	0	0	20	0
tlum	0	0	0	0	0	-30	30	0
Sum	0	0	0	0	0	0	0	

3.2. 記号

内生変数

記号	説明	プログラム内での記号
Y_i	生産水準	y(i)
U	効用水準	u
G	政府支出	g
fd_{fi}	生産要素への需要	fd(f,i)
cd_i	消費需要	cd(i)
gd_i	政府支出需要	gd(i)
p_i^Y	財の価格	py(i)
p_i^{VA}	VA の価格指数	pva(i)
p^U	効用の価格	pu
p^G	政府支出の価格	pg
p_f^F	生産要素の価格	pf(f)
c_i^Y	生産の単位費用	cy(i)
c^U	効用の単位費用	cu
c^G	政府支出の単位費用	cg
INC^H	家計の所得	inc_hh
INC^G	政府の所得	inc_gov
D^{obj}	ダミーの変数	dummy_obj

外生変数

記号	説明	プログラム内での記号
t^C	消費税率	tc
t_i^Y	生産税率 (グロスベース)	ty(i)
T^{LUM}	一括税	tlum
E_f^F	要素賦存量	end(f)

パラメータ

記号	説明	プログラム内での記号
σ_i^F	生産要素間の代替の弾力性	eos_f(i)
σ^C	消費における代替の弾力性	eos_c

θ_{ij}^{INT}	中間投入財のシェア	`sh_int(i,j)`
θ_{i}^{VA}	VA のシェア	`sh_va(i)`
θ_{fi}^{F}	VA における K と L のシェア	`sh_f(f,i)`
θ_{i}^{C}	消費における各財のシェア	`sh_c(i)`
θ_{i}^{G}	政府支出に占める各財のシェア	`sh_g(i)`

3.3. 単位費用と価格指数の定義

以下、モデル（均衡条件式）を数式によって表現していく。

- バー付きの記号はその変数のベンチマーク・データにおける値を表す。
- プログラム内ではベンチマークに 1 をとる値については省略している部分が多い。例えば、$\bar{p}_i^{VA} = \bar{p}_f^{F} = 1$と仮定しているので、VA の価格指数はプログラム内では次のような表現になっている。

$$p_i^{VA} = \left[\sum_f \theta_{fi}^{F} \left(p_f^{F}\right)^{1-\sigma_F} \right]^{\frac{1}{1-\sigma_F}}$$

VA の価格指数

$$p_i^{VA} = \bar{p}_i^{VA} \left[\sum_f \theta_{fi}^{F} \left(\frac{p_f^{F}}{\bar{p}_f^{F}} \right)^{1-\sigma_F} \right]^{\frac{1}{1-\sigma_F}} \qquad \{p_i^{VA}\}$$

生産の単位費用

$$c_i^{Y} = \bar{c}_i^{Y} \left[\sum_j \theta_{ji}^{INT} \left(\frac{p_j^{Y}}{\bar{p}_j^{Y}} \right) + \theta_i^{VA} \left(\frac{p_i^{VA}}{\bar{p}_i^{VA}} \right) \right] \qquad \{c_i^{Y}\}$$

効用の単位費用

$$c^{U} = \bar{c}^{U} \left[\sum_i \theta_i^{C} \left(\frac{p_i^{Y}}{\bar{p}_i^{Y}} \right)^{1-\sigma_C} \right]^{\frac{1}{1-\sigma_C}} \qquad \{c^{U}\}$$

政府支出の単位費用

$$c^{G} = \bar{c}^{G} \left[\sum_i \theta_i^{G} \left(\frac{p_i^{Y}}{\bar{p}_i^{Y}} \right) \right] \qquad \{c^{G}\}$$

3.4. 需要関数の定義

生産要素への需要

$$fd_{fi} = \overline{fd}_{fi}\left[\frac{p_i^{VA}/\bar{p}_i^{VA}}{p_f^F/\bar{p}_f^F}\right]^{\sigma_i^F} \qquad \{fd_{fi}\}$$

消費需要

$$cd_i = \overline{cd}_i\left[\frac{c^U/\bar{c}^U}{p_i^Y/\bar{p}_i^Y}\right]^{\sigma^C} \qquad \{fd_{fi}\}$$

政府支出需要

$$gd_i = \overline{gd}_i \qquad \{gd_i\}$$

3.5. ゼロ利潤条件（利潤最大化条件）

生産水準

$$c_i^Y = \frac{1-t_i^Y}{1-\bar{t}_i^Y}\ p_i^Y \qquad \{Y_i\}$$

効用水準

$$\frac{1+t^C}{1+\bar{t}^C}c^U = p^U \qquad \{U\}$$

政府支出

$$c^G = p^G \qquad \{G\}$$

3.6. 市場均衡条件

財の価格

$$Y_i = \sum_j INT_{ij}Y_j + cd_iU + gd_iG \qquad \{p_i^Y\}$$

生産要素の価格

$$E_f^F = \sum_i fd_{fi}Y_i \qquad \{p_f^F\}$$

効用の価格

$$U = INC^H/p^U \qquad \{p^U\}$$

政府支出の価格

$$G = INC^G/p^G \qquad \{p^G\}$$

3.7. 所得の定義

家計の所得

$$INC^H = \sum_f p_f^F E_f^F - p^U T^{LUM} \qquad \{INC^H\}$$

政府の所得

$$INC^G = \sum_i t_i^Y p_i^Y Y_i + t^C \sum_i p_i^Y cd_i U + p^U T^{LUM} \qquad \{INC^G\}$$

ダミーの目的関数（NLP モデル用）

$$D^{obj} = 1 \qquad \{D^{obj}\}$$

4. GAMS における変数、式の値の意味

後のデバッグにおいて、「変数の値」、あるいは「式の値」の情報を利用してデバッグをしていく。そこでまず、GAMS における「変数の値」、「式の値」について説明をしておく。

4.1. 変数に対する値

GAMSでは各変数（variableで宣言される変数）に対して以下の「4つの値」が関連付けられている[2]。

- 「LOWER」、「UPPER」、「LEVEL」、「MARGINAL」
- 変数名をxとすると、LOWER値、UPPER値、LEVEL値、MARGINAL値はそれぞれx.lo、x.up、x.l、x.mで表される。

各値の意味

- LOWER値：これは変数の「下限値」を表す。
- UPPER値：これは変数の「上限値」を表す。
- LEVEL値：これが普通の意味での変数の値であり、その変数がその時点でとっている値を示す。
- MARGINAL値：これは変数の「marginal value or dual value」を表す。この値が表す意味は、モデルのタイプによって変わってくる。

```
                    LOWER      LEVEL      UPPER    MARGINAL
---- VAR X            .         0.916      +INF       .
---- VAR Y            .         0.993      +INF       .
---- VAR U            .         0.798      +INF       .
```

[2] 実際には、この 4 つの値以外もある。

例えば、上のような計算結果となっているとすると「変数Xの下限値は0、上限値は無限大、MARGINAL値は0、値は0.916」ということ。

□ MARGINAL値

- MCPのモデルでは、MARGINALの値はその変数が関連付けられた条件式における乖離幅を表す。

［注］MCPのモデルを宣言・定義するときには、式だけではなく、その式に対応する変数を指定する[3]。例えば、サンプルのプログラム内でmodel_mcpというモデルは次のように定義されている。

```
*       MCPのモデル
model model_mcp Model by MCP /
      e_pva.pva,
      e_cy.cy,
      e_cu.cu,
      e_cg.cg,
      e_fd.fd,
      e_cd.cd,
      e_gd.gd,
      e_y.y,
      e_u.u,
      e_g.g,
      e_py.py,
      e_pf.pf,
      e_pu.pu,
      e_pg.pg,
      e_inc_hh.inc_hh,
      e_inc_gov.inc_gov
       /;
```

例えば、「e_pva.pva」というのは、e_pvaという式に対して、pvaという変数が関連づけ

[3] 厳密には、MCP のモデルであっても、変数の定義域に何も制約を課さないのなら、NLP のモデルのように式を列挙するだけでよいが、経済学のモデルでは変数の定義域に何らかの制約（非負制約など）を課すことが普通である。定義域に制約がある変数については、本文の説明のように、その変数に対応する式を指定しなければならない。

られているという指定である。

例：変数pがあり、それは以下のように定義される式e_pに関連付けられているとする。

```
Eq_p .. a – b*p =e= c + d*p;
```

この場合、p の MARGINAL 値（p.m）は

$$p.m = a – b*p - (c + d*p)$$

となる。

例：同様に、変数pgが、次式に関連付けられているとする。

```
e_pg .. g0 * g =e= inc_gov / pg;
```

このとき

$$pg.m = g0*g – inc\_gov/pg$$

である。

以上のように、MCP モデルにおける変数の MARGINAL 値はその変数が関連づけられた式の乖離幅を表す。これは **MARGINAL 値が非ゼロなら、式が等号で満たされていないということ**を意味することになる。

4.2. 式に対する値

式に対しても LEVEL、LOWER、UPPER の値がある（MARGINAL 値もあるが、以下では利用しないので省略）。

	LOWER	UPPER	LEVEL
=e=	rhs	rhs	変数パートの値
=l=	-inf	rhs	変数パートの値
=g=	rhs	Inf	変数パートの値

- LEVEL 値が表すのは常にその式の**「変数パート」**の値である。
  - 変数パートとは（variable 命令で宣言される）内生変数を含む部分のことである。
  - 変数パートの値は内生変数を含む部分を全て式の**左辺に移項した形**で表現される。
- 一方、LOWER 値、UPPER 値が表す値は式において「=e=」、「=l=」、「=g=」のど

れが利用されているかで変わってくる。上の表にまとめている。

- 「rhs」は**「定数パート」**の値を表す
  - 定数パートとは定数（parameter や scalar で定義されるもの）を含む部分のことである。
  - 定数パートの値は定数を含む部分を全て**式の右辺に移項した形**で表現される。

例 1:

```
    e_1 .. x + 2*y + 10 =e= 30;
```

において、x と y は変数であり、x.l=20、y.l=5 であるなら、

```
e_1.l = 20 + 2*5 = 30
e_1.lo = 30 – 10 = 20
e_1.up = 30 – 10 = 20
```

```
                        LOWER     LEVEL     UPPER    MARGINAL

---- EQU e_1           20.000    30.000    20.000       .     INFES
```

となる。

例 2

```
    e_2 .. x  =e=  y * (a / z)**(b);
```

で、x、y、z は変数、a と b はパラメータ、

```
x.l = 20
y.l = 2
z.l = 2
a = 6
b = 2
```

であるなら

```
e_2.l = 20 - 2*(6/2)**(2) = 2
e_2.lo = 0
```

```
e_2.up = 0
```

```
                          LOWER      LEVEL      UPPER    MARGINAL

---- EQU e_2                .         2.000       .         .    INFES
```

となる（定数パートはないので UPPER と LOWER はゼロ）。

例 3

```
     e_3 .. a*x +  b*y =l= c;
```

で、x と y は変数、a、b、c はパラメータで

```
x.l = 10
y.l = 20
a = 6
b = 2
c = 100
```

であるなら

```
e_3.l = 6*10 + 2*20 = 100
e_2.lo = -inf
e_2.up = c = 100
```

```
                          LOWER      LEVEL      UPPER    MARGINAL

---- EQU e_3             -INF      100.000    100.000       .
```

となる（=l=が利用されているので、LOWER は「-inf」となる）。

□ 式に対する値（まとめ）

- 式に対する値の定義より、式が等号で満たされているなら
  - LEVEL 値と LOWER 値が等しい
  - LEVEL 値と UPPER 値が等しい

の少なくともどちらか一方が満たされていないといけない。

- 言い換えれば、LEVEL 値が LOWER 値とも UPPER 値とも異なっているのなら、式は等号では満たされていない（つまり、式の右辺と左辺は等しくない）ということになる。

5. Benchmark replication によるデバッグ

CGE 分析は、ベンチマーク・データの下で均衡状態が成立しているという前提でおこなわれる。Benchmark replication によるデバッグとは、実際にその均衡状態となっているかをチェックすることで、モデル、データのバグを見つける方法である。

5.1. MCP モデルのケース

まず、MCP で記述したモデルによって説明する。how-to-debug-mcp.gms を例に使い説明する。benchmark replication とは具体的には次のような形でモデルを実行することである。

```
model_mcp.iterlim = 0;
solve model_mcp using mcp;
```

つまり、iteration の回数をゼロに設定してモデルを解くということである。iteration の回数をゼロとして解くとは、変数の初期値（ベンチマーク・データ）が均衡条件（連立方程式）を満たしているか否かを判断して計算を終了するということである。

これでモデルが正常に解けるなら、ベンチマーク・データの下で均衡が成立しているという前提が満たされていることになるが、正常に解けていないのならモデルかデータのどこかにバグが存在するということになる。

実際に、how-to-debug-mcp.gms を解いてみると、次のような結果となる。

```
                           LOWER      LEVEL      UPPER     MARGINAL

---- VAR u                    .        1.000      +INF       0.022
---- VAR g                    .        1.000      +INF        .
```

変数uのMARGINAL値が0になっていない。これは変数uが関連づけられた式が等号では満たされていないことを意味する。変数uはモデルの宣言におけるe_u.uという指定により、式e_uに関連づけられているので、e_uが等号で満たされていないということになる。

式e_uの定義をチェックすると

```
    (1 + tc) * cu =g= pu;
```

となっているが、cu、puともにベンチマーク値が1であるので、上式は等号では満たされない。このため右辺左辺に差が生じ、MARGINAL値が0にならないことがわかる。上の式は

```
    (1 + tc) * cu / (1+tc0) =g= pu;
```

の間違いであるので、修正して解き直すと

```
                          LOWER      LEVEL      UPPER     MARGINAL

---- VAR u                  .        1.000      +INF        .
---- VAR g                  .        1.000      +INF        .
```

となり、MARIGNAL値はゼロになる。

基本的には以上の作業を繰り返せばよい。

- MARGINAL値がゼロになっていない変数を探す。
- それは、その変数が対応する式が等号では満たされていないことを意味する。
- 式をチェックして修正する。

e_uを修正し解きなおし、さらに他の変数をチェックする。

```
---- VAR fd  生産要素への需要

          LOWER      LEVEL      UPPER     MARGINAL

k.agr       .        70.000      +INF        .
k.man       .       140.000      +INF      20.000
k.ser       .       100.000      +INF        .
l.agr       .        70.000      +INF        .
l.man       .        90.000      +INF      20.000
l.ser       .       300.000      +INF        .
```

部門manの変数fdの MARGINAL値が非ゼロになっている。変数fdに対応するのは、式e_fdで、これは要素需要の定義式である。式e_fdの右辺左辺の値をチェックするため、次のようなparameterを定義し、計算をしてみる（solve命令のすぐ前に以下のコードを置く）。

```
Parameter
    chk_fd;
chk_fd(f,i,"l") = fd.l(f,i);
chk_fd(f,i,"r") = fd0(f,i) * (pva.l(i) / pf.l(f))**(eos_f(i));
display chk_fd;
```

結果は次のようになる。

```
----    533 PARAMETER chk_fd

                l           r

k.agr      70.000      70.000
k.man     140.000     120.000
k.ser     100.000     100.000
l.agr      70.000      70.000
l.man      90.000      70.000
l.ser     300.000     300.000
```

左辺・右辺ともベンチマークにおける要素需要量になっていなければならないが、以上のように左辺の値がおかしいことがわかる。

そこで、fdの初期値を設定している部分をチェックすると

```
fd.l(f,i) = fd0(f,i) + err3(i);
```

のように式がおかしいことがわかる。これを次の正しい定義に修正する。

```
fd.l(f,i) = fd0(f,i);
```

再びモデルを解き、変数をチェックする。

```
---- VAR fd  生産要素への需要

         LOWER     LEVEL     UPPER    MARGINAL
```

```
k.agr       .       70.000     +INF        .
k.man       .      120.000     +INF        .
k.ser       .      100.000     +INF        .
l.agr       .       70.000     +INF        .
l.man       .       70.000     +INF        .
l.ser       .      300.000     +INF        .
```

今度はMARGINAL値が0になっていることがわかる。

次

```
---- VAR py  財の価格

       LOWER     LEVEL     UPPER    MARGINAL

agr 1.0000E-6     1.000      +INF   -130.000
man 1.0000E-6     1.000      +INF   -130.000
ser 1.0000E-6     1.000      +INF   -310.000
```

変数pyのMARGINAL値がどの財についても非ゼロになっている。変数pyが対応する式e_pyを見てみると

```
e_py(i) ..
    y0(i) * y(i) =g= sum(j, int0(i,j) * y(j)) + cd(i) * u + gd(i) * g;
```

この式は左辺が財iの供給量、右辺が財iの需要量を表しており、財iの市場均衡条件になっている。この式の左辺、右辺の値をチェックしみる。

```
Parameter
    chk_py;

chk_py(i,"l") = y0(i) * y.l(i);
chk_py(i,"r") = sum(j, int0(i,j) * y.l(j)) + cd.l(i) * u.l + gd.l(i) * g.l;
display chk_py;
```

というコードをsolve命令の後に追加し、chk_pyの中身を確認する。

```
----    540 PARAMETER chk_py

              l           r

agr     200.000     330.000
man     300.000     430.000
ser     500.000     810.000
```

e_pyの右辺の値がおかしいことがわかる。右辺の各項目の値を確かめると、gd.l(i)*g.lの値（政府支出のための需要）がおかしいことがわかる。gd.l(i)の中身を追っていくと、結局、

```
gd0(i) = - sam(i,"u");
```

という部分がおかしいことがわかる。これを正しい

```
gd0(i) = - sam(i,"g");
```

に修正する。すると、pyのMARGINAL値はゼロになる。

次

```
---- VAR pf  生産要素の価格

     LOWER     LEVEL     UPPER    MARGINAL

k 1.0000E-6     1.000     +INF   -580.000
l 1.0000E-6     1.000     +INF   -880.000
```

変数pfのMARGINAL値が非ゼロである。pfに対応するe_pf式をチェックする。

```
e_pf(f) .. end(f) =g= sum(i, fd(f,i) * y(i));
```

左辺、右辺の中身をチェックするためにparameterを定義。

```
parameter chk_pf;
chk_pf(f,"l") = end(f);
chk_pf(f,"r") = sum(i, fd.l(f,i) * y.l(i));
```

```
display chk_pf;
```

結果

```
----    545 PARAMETER chk_pf

            l           r

k    -290.000     290.000
l    -440.000     440.000
```

生産要素の供給量（賦存量）がマイナスになっており、明らかにおかしい。end(f)の定義の部分をチェックする。

```
end0(f) = - sam(f,"hh");
```

がおかしいことがわかる。次のように正しく修正する。

```
end0(f) = sam(f,"hh");
```

次

```
---- VAR cy  生産の単位費用

       LOWER     LEVEL     UPPER    MARGINAL

agr 1.0000E-6     1.000      +INF       0.663
man 1.0000E-6     1.000      +INF       0.611
ser 1.0000E-6     1.000      +INF       0.766
```

生産の単位費用のMARIGNAL値が非ゼロ。

```
e_cy(i) ..
    cy(i) =e=
    sum(j, sh_int(j,i) * py(j)) + sh_va(i) * (pva(i)/err1(i));
```

err1(i)がおかしい。修正

```
e_cy(i) ..
    cy(i) =e= sum(j, sh_int(j,i) * py(j)) + sh_va(i) * pva(i);
```

次

```
                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR cu          1.0000E-6     1.000     +INF       .
---- VAR cg          1.0000E-6     1.000     +INF       .
---- VAR inc_hh      1.0000E-6   700.000     +INF       .
---- VAR inc_gov     1.0000E-6   110.000     +INF      5.000
```

政府の所得変数（inc_gov）のMARINAL値が非ゼロ。inc_govの定義をチェック。

```
e_inc_gov ..
    inc_gov =e=
    sum(i, ty(i) * py(i) * y0(i) * y(i))
    + tc * (sum(i, py(i) * cd(i) * u)) + pu * tlum;
```

parameterを定義して、右辺、左辺の値をチェック。

```
parameter  chk_inc_gov;
chk_inc_gov("l") = inc_gov.l;
chk_inc_gov("r") = sum(i, ty(i) * py.l(i) * y0(i) * y.l(i))
    + tc * (sum(i, py.l(i) * cd.l(i) * u.l)) + pu.l * tlum;
display chk_inc_gov;
```

結果

```
----    533 PARAMETER chk_inc_gov

l 110.000,    r 105.000
```

右辺がおかしい。さらに右辺の中身を細かくチェック。

```
parameter  chk_inc_gov;
chk_inc_gov("l") = inc_gov.l;
```

```
chk_inc_gov("r") = sum(i, ty(i) * py.l(i) * y0(i) * y.l(i))
    + tc * (sum(i, py.l(i) * cd.l(i) * u.l)) + pu.l * tlum;
chk_inc_gov("r1") = sum(i, ty(i) * py.l(i) * y0(i) * y.l(i));
chk_inc_gov("r2") = tc * (sum(i, py.l(i) * cd.l(i) * u.l));
chk_inc_gov("r3") = pu.l * tlum;
display chk_inc_gov;
```

結果

```
----    555 PARAMETER chk_inc_gov

l  110.000,    r  105.000,    r1  60.000,    r2  15.000,    r3  30.000
```

消費税収を表すr2の部分がおかしい。消費税収は本当は20であるはず。消費税率tc0の定義のチェック。

```
tc0 = - (sam("tc","u") + err2) / sum(i, cd0(i));
```

ここが間違い。以下のように修正する。

```
tc0 = - sam("tc","u") / sum(i, cd0(i));
```

Iterationの回数を0に設定にした上で、以下の作業を繰り返す。

- モデルを解き、MARGINAL値がゼロになっていない変数を探す
- それはその変数が対応する式が等号では満たされていないことを意味する
- 式をチェックして修正する

5.2. NLP モデルのケース

MCP モデルにおいては次のような手順に従った。

- 変数の MARGINAL 値はその変数に関連づけられた式の左辺と右辺の差（乖離）を表す。
- 変数の MARGINAL 値をチェック
- MARGINAL 値がゼロでない変数に関連づけられた式をチェック

NLP モデルであっても基本的な考え方は MCP モデルと同じであり、式が等号で満たされているかどうかをチェックすることで、間違いを見つけていく。しかし、NLP のモデルの場合には、MCP モデルと違い変数と式を関連づけていないため、直接式をチェックする

という方法をとる。

第 4.2 節で見たように、式については、その LEVEL 値、LOWER 値、UPPER 値の値を比較することで、式が等号で満たされているかどうかをチェックすることができる。つまり、LEVEL 値が LOWER 値とも UPPER 値も等しくなければ、式は等号で満たされていないことになる。

例

```
---- EQU e_y  生産水準

       LOWER     LEVEL     UPPER    MARGINAL

agr        .         .        +INF       EPS
man        .         .        +INF       EPS
ser        .         .        +INF       EPS
```

では LOWER 値と LEVEL 値がともにゼロ（ピリオドはゼロを表す）で等しいので、式は等号で満たされている。

```
---- EQU e_gd  政府支出需要

       LOWER     LEVEL     UPPER    MARGINAL

agr   150.000   150.000   150.000       EPS
man   180.000   180.000   180.000       EPS
ser   350.000   350.000   350.000       EPS
```

この式についても、LOWER 値、LEVEL 値、UPPER 値が全て等しいので、式は等号で満たされている。

```
                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU e_u             .        0.022     +INF       EPS
---- EQU e_g             .         .        +INF       EPS
```

これについても、e_g は LOWER 値と LEVEL 値がゼロで等しいので、等号で満たされて

いる。一方、e_u については LEVEL 値が LOWER 値とも UPPER 値とも異なるので、等号で満たされていない。

```
                        LOWER      LEVEL      UPPER    MARGINAL

---- EQU e_cu             .          .          .         EPS
---- EQU e_cg             .       -5.182        .         EPS  INFES
---- EQU e_inc_hh         .     1460.000        .         EPS  INFES
---- EQU e_inc_gov        .        5.000        .         EPS  INFES
---- EQU e_dummy_o~     1.000      1.000      1.000       EPS
```

e_cg、e_inc_hh、e_inc_gov はどれも LEVEL 値が LOWER 値、UPPER 値と異なるので、やはり等号では満たされていない。

それでは、実際に、how-to-debug-nlp.gmsの出力を読んでいこう。まず、式e_uを見ると等号で満たされていない。

```
                        LOWER      LEVEL      UPPER    MARGINAL

---- EQU e_u              .         0.022       +INF       EPS
---- EQU e_g              .          .          +INF       EPS
```

よって、e_uの部分がおかしいということになる。

同様に式をチェックしていくと、

```
---- EQU e_fd  生産要素への需要


          LOWER      LEVEL      UPPER    MARGINAL

k.agr       .          .          .         EPS
k.man       .       20.000        .         EPS  INFES
k.ser       .          .          .         EPS
l.agr       .          .          .         EPS
l.man       .       20.000        .         EPS  INFES
```

```
1.ser      .          .          .          EPS
```

というように、e_fdの一部が等号で満たされていないことがわかる。よって、e_fdをチェックすればよい。

あとは、MCPモデルと同じように修正していけばよい。

6. 参考文献

- GAMS Development Corporation, (2010) *GAMS: A User's Guide,* 2010.
- McCarl, Bruce A., Alex Meeraus, Paul van der Eijk, Michael Bussieck, Steven Dirkse and Pete Steacy, *McCarl GAMS User Guide: Version 23.3*.