成城大学経済研究所
研究報告 No. 34

Mathematica によるミクロ経済学

小　平　裕

2002年4月

The Institute for Economic Studies
Seijo University

6–1–20, Seijo, Setagaya
Tokyo 157-8511, Japan

Mathematica によるミクロ経済学

小　平　　裕

1. はじめに

本稿の目的は，Mathematica の基本的な操作を説明し，経済学のどのような問題に Mathematica を使うことができるか，どのように使えば有効かを，ミクロ経済学の例を通じて検討することである。われわれの関心は経済学における Mathematica の有効な利用法にあるので，本稿で取り上げる Mathematica の操作は経済学に必要な最小限の基本的機能に限定される。

経済学では，一般均衡理論，数理ファイナンスや計量経済学のように数学を多用する分野だけではなく，基本的な分野においても数学は利用されている。例えば，ミクロ経済学において消費者の効用最大化行動から需要関数を，生産者の利潤最大化行動から供給関数を導出する際や，市場の需要関数と供給関数から均衡価格を求める際に数式を解くこと，あるいはその様子を理解するためにグラフを用いることは標準的な手法である。

しかし，経済学の初学者の中には数学や計算を苦手とする人も多く，このことが経済学学習の障害となっている。もし容易に数式を解いたり，グラフを描いたりすることができるならば，経済学の学習に専念できることになろう。本研究の成果は，学部教育の向上に利用可能であるばかりでなく，今後増えることが期待される社会人入試による大学院生の教育にも役立とう。

数式を解いたり，グラフを描いたりすることができるコンピュータ・ソフト

ウェアとして Mathematica[1] がある。Mathematica は汎用の数式処理ソフトウェアの１つである。ここで数式処理とは，例えば「$\sin x$ を微分せよ」という入力に対して「$\cos x$」という結果を出力する文字式を処理する情報処理のことであり，加減乗除を行う数値計算とは全く異なる計算機の利用形態である。この Mathematica は，それ以前から知られていた数式処理ソフトウェア REDUCE, MACSYMA, muMATH などに比べて処理能力が高い上に，数値計算も容易であり，強力なグラフ描画機能を持っているので，現在では他よりも広く普及している。すなわち，Mathematica は記号処理，数値処理，グラフ処理という数学の３つに大別される計算処理を統一的な方法で取り扱うことができる総合的なソフトウェアであり，定量的な分析が行われるあらゆる分野で利用されている。

2. Mathematica の操作

2.1 Mathematica はインタプリタ型（即時実行形式）ソフトウェアである。私たちが何か尋ねると，Mathematica はその都度その命令を解釈して実行するという会話形式で利用する。Mathematica の長所は，予め多くの関数（指令，コマンドのためのプログラム）が組み込み関数として準備されており，利用者は目的に応じてそれらを組み合わせて使用できることにあるといえよう。したがって，初めて Mathematica を利用しようとする者は組み込み関数の多さに圧倒され，敷居が高いと感じるかも知れない。実際，「システム開発者によるオフィシャル・ガイド」と名付けられた Steven Wolfram の『Mathematica ブック（改訂第３版）』（Wolfram（1996，榊原，武沢ほか訳 (1998)）は1300ページを超えており[2]，初心者はその分厚さに戸惑うかも知れないが，経済学の研究で使用する組み込み関数はそれほど多くはない。

本節では Mathematica を全く利用したことのない読者を想定して，本稿を読むにあたって必要な Mathematica の使い方について最小限の知識を解説する。

1） Version 1.0 は1988年６月に発表された。Macintosh，Windows などのパソコンをはじめ，各種のワークステーション（Linux や各種の Unix）で利用可能である。最新は version 4.1 である（2002年２月現在）。

2） 最新の version 4 には，Wolfram (1996, 榊原，武沢訳 (1998)) に加えて別冊の『Mathematica ブック追加項目集』（149ページ）が付いている。version 2 のマニュアル（Wolfram（1991，白水訳 1992））は，937ページであった。

ミクロ経済学における Mathematica の具体的な利用法については，本節の初歩的な利用法の説明を踏まえて，第3節，第4節で検討する。経済学における利用例を見れば，本節で説明した少数の基本的操作法を繰り返し組み合わせているだけであることが明らかとなるであろう。

2.2 数値計算（加減乗除）の操作法の説明から始めよう。足し算には（筆算と同じく）＋（プラス）を，引き算には－（マイナス）を使う。Mathematica を起動すると表れる何も書かれていない白い入力画面（ノートブック notebook と呼ばれる）に，ワープロのつもりで直接入力により例えば次の式を書いてみよう。

```
123 + 45
```

いま入力したばかりの「123+45」は，ノートブックには太字 Bald で表示される。なお，数字や数学記号は半角でなければならない。

Mathematica にこの入力を解釈し処理して，結果を出させるには，shift キーを押しながら enter キーを押す（以下では，この操作を shift+enter と表す）[3]。enter キーだけを押すと改行され，入力が2行以上にわたることになる[4]。shift+enter とキーを押すと，入力行の行頭には入力配列を意味する In [ ] : = が付き，同時に入力行の下に出力配列を示す Out [ ] = が表れ，その右側に 168 という結果が現れる（出力は太字ではなく標準の立体）。In や Out の後の [ ]（角カッコ brackets）の中の数字は，実行順を示す通し番号であり，Mathematica により自動的に付される。また，各行の右側にはセルの範囲を示す記号セルブラケットが付けられる。

```
In[1]:= 123 + 45

Out[1]= 168
```

この一連のプロセスを（コマンドを）「評価する」evaluate という。

引き算も同様である。ノートブック画面には直前の結果 Output の下に水平線が引かれているので，その下にカーソルを合わせて，直接入力により次の式

3) 以上は，Windows 版の場合の操作である。Macintosh 版の場合には，shift キーを押しながら return キーを押す (shift+return) か，あるいはテンキー部にある enter キーを単独で押すかすれば良い。

4) Macintosh 版の場合には，return キーだけを押すと改行され，2行以上にわたる入力が可能になる。

を挿入する。前と同様に shift+enter とキーを押せば，すなわち評価すれば，その行の下に 78 という結果が現れる。

```
In[2]:= 123 - 45
Out[2]= 78
```

2.3　掛け算には，×の代わりに＊（アスタリスク）を用いる。例えば，掛け算123×45の積を求めるには，

```
In[3]:= 123 * 45
Out[3]= 5535
```

とする。なお，２つの数字の間を１字分の空白（スペース）を入れても，Mathematica は掛け算のコマンドであると認識する。勿論，結果は変わらない。

```
In[4]:= 123 45
Out[4]= 5535
```

また，version 3 以降の Mathematica であれば，入力にパレット Palettes を利用することもできる。具体的は，ファイル File をクリックしてメニューを開き，そこからパレットを選ぶ。表れるサブメニューの中から「基礎的な入力」BasicInput を選択すると，ノートブックの外に数式エディタ風のパレットが開くから，その中から×を選ぶこともできる。

```
In[5]:= 123 × 45
Out[5]= 5535
```

割り算には，÷の代わりに /（スラッシュ）を使う。例えば，割り算123÷45の商を求めるには，

```
In[6]:= 123 / 45
Out[6]= 41/15
```

とする。version 3 以降であれば，割り算でも基礎的な入力パレットを開いて，÷を選ぶこともできる。

```
In[7]:= 123 ÷ 45
Out[7]= 41/15
```

あるいは，基礎的な入力パレットを使って，分数の形で入力しても同じ結果を得る。

```
In[8]:= 123/45
Out[8]= 41/15
```

2.4　ここで，上のように 123 ではなく，小数点付きの 123. すなわち 123.0 を 45 で割るという計算を行うと，結果は違ってくる。

```
In[9]:= 123. / 45
Out[9]= 2.73333
```

その理由は，Mathematica では 123 は厳密値と見なされるのに対して，小数点付きの値は一定の精度（規定値の機械精度では 6 桁）の近似値と見なされ，式の中に近似値が含まれると評価も近似値で行われるからである。小数点の付かない 123 を 45 で割った商（結果）の近似値を得るには，組み込み関数 N を用いる。組み込み関数を表現するには，最初の文字を必ず大文字で始め（この組み込み関数 N は，関数名がたまたま 1 文字である），[　] （角カッコ）を用いる。

```
In[10]:= N[123 / 45]
Out[10]= 2.73333
```

あるいは，//N（2 重スラッシュの後に組み込み関数名の N，大文字）を付けても，同じ結果が得られる[5]。

```
In[11]:= 123 / 45 // N
Out[11]= 2.73333
```

5)　組み込み関数には 4 つの表記法があり，N [*expr*] は標準形，*expr*//N は後置形と呼ばれる。詳しくは，Wolfram (1996, sec. 2.1.3)を参照せよ。

ここで，表現 *expr* の *n* 桁の近似値を求めるには，組み込み関数 NumberForm [N[*expr*], *n*] を使う[6]。例えば，次のようにすれば，割り算 123÷45 の商の 20 桁の近似値が得られる。

```
In[12]:= NumberForm[N[123 / 45], 20]

Out[12]//NumberForm=
        2.7333333333333333
```

2.5　べキ乗は変数の肩に書く代わりに，^（ハット）を使う。例えば，2 の 3 乗，つまり 2^3 は次のようにして求める。

```
In[13]:= 2^3

Out[13]= 8
```

なお，基礎的な入力パレットが使えれば，次のようにしても良い。

```
In[14]:= 2^3

Out[14]= 8
```

以上から分かるように，数値の計算（加減乗除とべキ乗）は，電卓の使い方とほぼ同じである。

最後に，積と商は和と差よりも優先される，（　）（丸カッコ paretheses）内の計算が優先されるなどの計算の優先順序に関する法則は，通常の算術演算と同じである。

```
In[15]:= 1 + 2*3

Out[15]= 7

In[16]:= (1 + 2)*3

Out[16]= 9
```

上の例では，入力配列 In[15] は 1 と 2×3 の和を求める式であり，1+6 という計算を行っている（答は 7）のに対して，入力配列 In[16] は，1 と 2 の和と 3

6）*n* を指定しない場合には，6 桁（機械精度）の近似値が求められる。なお，version 3 までは，N[*expr*, *n*] で表現 *expr* の *n* 桁の近似値が求められた。version 4 より，組み込み関数 N の機能が変更された。

の積を求める式であり，3×3 を計算している（答は 9）。

2.6　代数式（文字式）の表し方も，数字だけの場合と変わらない。例えば，x の 2 次式 $2x^2+3x+1$ は

```
In[17]:= 2*x^2 + 3*x + 1
Out[17]= 1 + 3 x + 2 x^2
```

と表される。勿論，＊（アスタリスク）の代わりに空白を入れても，あるいは基礎的な入力パレットを使っても良い。

1 つの代数式には幾通りもの表し方があるので，場合に応じて代数式を因数分解や展開する必要が生じる。多項式の因数分解には，組み込み関数 Factor を使う。上の 2 次式 $2x^2+3x+1$ を実際に因数分解してみよう。

```
In[18]:= Factor[%]
Out[18]= (1 + x) (1 + 2 x)
```

ここで，入力配列 In[18] の％（パーセント記号）は直前の結果の引用を意味し，この場合には多項式を $2x^2+3x+1$ を表している。つまり，In[18] は

```
Factor[2*x^2 + 3*x + 1]
```

とするのと同じである。なお，k 回前の結果を引用するには，％を k 個並べて $\overbrace{\%\ldots\%}^{k}$ とするか，$\%k$ とする。

逆に，多項式の展開には組み込み関数 Expand を使う。ここで $(x+1)(2x+1)$ を展開し，$2x^2+3x+1$ が得られることを確認しておこう。

```
In[19]:= Expand[%]
Out[19]= 1 + 3 x + 2 x^2
```

2.7　ベクトルや行列を表すには，複数の変数や式をまとめたリストを使う。例えば，3, 5, 6, 1, 2 という 5 つの数から構成されるリストを考えてみよう。リストは各要素を ｛ ｝（波カッコ braces）で括る[7]。これに A という名前を付

7)　カッコの使い方をまとめると，（　）（丸カッコ）は計算の優先順位を指定するために，｛ ｝（波カッコ）はリストの要素を示すために，［　］（角カッコ）は関数を表すために使わ

ける（リストを変数 A に割り当てる）には

```
In[20]:= A = {3, 5, 6, 1, 2}
Out[20]= {3, 5, 6, 1, 2}
```

とすれば良い。A に 2 を掛けた後，各要素に 1 を加えてみよう。

```
In[21]:= A*2 + 1
Out[21]= {7, 11, 13, 3, 5}
```

後の説明の都合上，これに B という名前を付けておこう。

```
In[22]:= B = %
Out[22]= {7, 11, 13, 3, 5}
```

リストから 1 つの要素（単一要素）を取り出す（リストの部分抽出）には，[[　]]（角カッコを 2 重にする）の中にその数字を書くか，組み込み関数 Part を使う。例えば，上の A の第 2 要素を取り出すには以下のようにコマンドを書く。

```
In[23]:= A[[2]]
Out[23]= 5
```

あるいは

```
In[24]:= Part[A, 2]
Out[24]= 5
```

リストから複数の要素を取り出すには，複数の要素番号を {　} （波カッコ）で括れば良い。例えば，A の第 2 要素と第 4 要素の 2 つを取り出すには，以下のようにすれば良い。

```
In[25]:= A[[{2, 4}]]
Out[25]= {5, 1}
```

れる。

あるいは

```
In[26]:= Part[A, {2, 4}]
Out[26]= {5, 1}
```

抽出順序も区別される。上の例に即していえば，第2要素と第4要素を取り出すことと，第4要素と第2要素を取り出すことは，違うコマンドと見なされる。

```
In[27]:= A[[{4, 2}]]
Out[27]= {1, 5}
```

リストの最初の n 個を取り出すには組み込み関数 Take を用いる。例えば，リスト A の最初の3個を取り出すには以下のようすれば良い。

```
In[28]:= Take[A, 3]
Out[28]= {3, 5, 6}
```

もし最後の2個を取り出したいのなら，-2 とすれば良い。

```
In[29]:= Take[A, -2]
Out[29]= {1, 2}
```

ベクトルはリストで表される。ベクトルの和と差は，リストの和と差で求められる。

```
In[30]:= A + B
Out[30]= {10, 16, 19, 4, 7}
In[31]:= A - B
Out[31]= {-4, -6, -7, -2, -3}
```

ベクトルの内積を求めるには，.（ピリオド）を用いて次のように入力する。

```
In[32]:= A.B
Out[32]= 167
```

行ベクトルと列ベクトルは，ともに Mathematica の構成上はリストなので，区

別されない。したがって，

```
In[33]:= B.A
Out[33]= 167
```

も可能であるので，注意が必要である。

当然，２つのベクトルの次元が一致していないと，内積は計算できない。ここでは A は５次元ベクトルであるので，例えば２次元ベクトル G を定義して，このことを確認しておこう。

```
In[34]:= G = {1, 1}
Out[34]= {1, 1}
In[35]:= A.G
        Dot::dotsh : テンソル{3, 5, 6, 1, 2}と{1, 1}は計算できない次元を持っています.
Out[35]= {3, 5, 6, 1, 2}.{1, 1}
In[36]:= G.A
        Dot::dotsh : テンソル{1, 1}と{3, 5, 6, 1, 2}は計算できない次元を持っています.
Out[36]= {1, 1}.{3, 5, 6, 1, 2}
```

なお，ベクトルの内積を求めるのに＊（アスタリスク）を使ってはならない。＊を使うと，次の例のように対応する要素同士の積を並べたベクトルを計算することになり，内積にはならないので注意せよ。

```
In[37]:= A*B
Out[37]= {21, 55, 78, 3, 10}
```

同じく，/（スラッシュ）を使うと，対応する要素の商を並べたベクトルが得られる。

```
In[38]:= A/B
Out[38]= {3/7, 5/11, 6/13, 1/3, 2/5}
```

なお，ここでも次元が一致していないと計算できない。

```
In[39]:= A * G
        Thread::tdlen : {3, 5, 6, 1, 2} {1, 1}で互いに長さが等しくない
        オブジェクト同士は結合できません.
Out[39]= {1, 1} {3, 5, 6, 1, 2}
In[40]:= A / G
        Thread::tdlen : {3, 5, 6, 1, 2} {1, 1}で互いに長さが等しくない
        オブジェクト同士は結合できません.
Out[40]= {1, 1} {3, 5, 6, 1, 2}
```

2.8 行列は，Mathematica ではリストのリストとして表される。例えば，3×2 の行列 M は，次のように2次元の行ベクトルを3つ重ねたものとして定義される。

```
In[41]:= M = {{a, b}, {c, d}, {g, h}}
Out[41]= {{a, b}, {c, d}, {g, h}}
```

この行列を行列形式で表示するには，組み込み関数 MatrixForm（M と F は大文字）を使う。

```
In[42]:= MatrixForm[M]
Out[42]//MatrixForm=
        a  b
        c  d
        g  h
```

あるいは，後置形を使って，//MatrixForm（2重のスラッシュの後に組み込み関数名）とする。

```
In[43]:= M // MatrixForm
Out[43]//MatrixForm=
        a  b
        c  d
        g  h
```

行列から行を取り出すには，組み込み関数 Part あるいは [[ ]]（2重の角カッコ）を使う。例えば，上の行列 M の第2行を取り出すには次の何れかを

行う。

```
In[44]:= Part[M, 2]
Out[44]= {c, d}
```

あるいは

```
In[45]:= M[[2]]
Out[45]= {c, d}
```

行列の第 i 行第 j 列の要素を取り出すにも [[　]]（2重の角カッコ）を用い，まず第 i 要素（行ベクトル）を，次にそのベクトルの第 j 要素（第 j 列）を取り出せばよい。例えば，行列 M の第1行第2列の要素 m_{12} を取り出すには，

```
In[46]:= M[[1, 2]]
Out[46]= b
```

あるいは，組み込み関数 Part を用い，

```
In[47]:= Part[Part[M, 1], 2]
Out[47]= b
```

とする。行列が何行何列かを調べるには，組み込み関数 Dimensions を用いる。行列 M に適用すると，

```
In[48]:= Dimensions[M]
Out[48]= {3, 2}
```

という結果が得られ，M は3行2列の行列であることが分かる。

行列とベクトルの積を求めるには，ベクトルの内積を求める場合と同様に，. （ピリオド）を使う（* （アスタリスク）ではない）。ここで，2要素のベクトル v を定義して，3×2 の行列 M の右から掛けてみよう。

```
In[49]:= v = {x, y}

Out[49]= {x, y}

In[50]:= M.v

Out[50]= {a x + b y, c x + d y, g x + h y}
```

ベクトルのところでも注意したように，Mathematica の上では行ベクトルと列ベクトルは区別されない。

勿論，行列とベクトルの次元が合わないと積は計算できない。例えば，3×2 の行列 M の左から 2 次元ベクトル v を掛けようとすると，エラーになる。

```
In[51]:= v.M

          Dot::dotsh : テンソル{x, y}と
              {{a, b}, {c, d}, {g, h}}は計算できない次元を持っています.

Out[51]= {x, y}.{{a, b}, {c, d}, {g, h}}
```

2.9　微分には組み込み関数 D を用いる。例えば，関数 x^3+4x^2+5x の x に関する微分 $\frac{d(x^3+4x^2+5x)}{dx}$ を求めるには，次のようにする。

```
In[52]:= D[x^3 + 4*x^2 + 5*x, x]

Out[52]= 5 + 8 x + 3 x^2
```

上で注意したように，ベキ乗は ^（ハット）を用いて表現される。パレットの中から基礎的な入力を開いて，通常の式の形を入力しても同じ結果が得られる。

```
In[53]:= D[x^3 + 4 x^2 + 5 x, x]

Out[53]= 5 + 8 x + 3 x^2
```

この関数の 2 回微分，すなわち $\frac{d^2(x^3+4x^2+5x)}{dx^2}$ を求めることは，さらに x を増やすことで実行される。

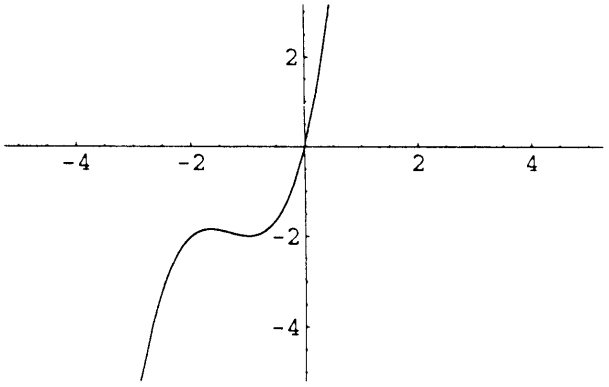
```
In[54]:= D[x^3 + 4 x^2 + 5 x, x, x]

Out[54]= 8 + 6 x
```

この関数 x^3+4x^2+5x のグラフを描いてみよう。この関数は 1 つの変数（説

明変数，独立変数とも呼ばれる，この場合は x）と関数値（非説明変数，従属変数とも呼ばれる）の関係を示しているので，グラフは2次元グラフになる。2次元グラフィックスを描くためには，組み込み関数 Plot を用いる。変数 x の定義域を［-5, 5］とした場合のグラフは，次のようなコマンドによって描くことができる。

```
In[55]:= Plot[x^3 + 4 x^2 + 5 x, {x, -5, 5}]
```




```
Out[55]= - Graphics -
```

関数を積分するには，組み込み関数 Integrate を用いる（注意，ここで I と書いてはならない。I は虚数単位を表す）。例えば，不定積分 $\int(x^3+4x^2+5x)dx$ を求めるには，

```
In[56]:= Integrate[x^3 + 4 x^2 + 5 x, x]
```

$$\text{Out[56]=}\quad \frac{5\,x^2}{2} + \frac{4\,x^3}{3} + \frac{x^4}{4}$$

とする。定積分 $\int_a^b(x^3+4x^2+5x)dx$ を求めるには，

```
Integrate[x^3+4x^2+5x, {x, a, b}]
```

とすればよい。

もし関数が多変数関数であれば，組み込み関数 D は偏微分を計算する。すなわち，偏微分にも組み込み関数 D を用いる。例えば，関数 $x^2+5x+2y^2+3y+2xy$ の x に関する偏微分 $\frac{\partial(x^2+5x+2y^2+3y+2xy)}{\partial x}$ を求めるのは，次のようにする。

```
In[57]:= D[x^2 + 5*x + 2*y^2 + 3*y + 2*x*y, x]

Out[57]= 5 + 2 x + 2 y
```

ここで，$x \times y$ のつもりで xy と書いてはならない。xy は1つの変数と認識される。$x \times y$ を入力するには，上述したように，(i) *（アスタリスク）を使って $x*y$ とするか，(ii) x と y の間にスペースを空けて $x\ y$ とするか，(iii) 基礎的な入力パレットを使って $x \times y$ とするかしなければならない。この結果をさらに y で偏微分するには，すなわち $\frac{\partial(x^2+5x+2y^2+3y+2xy)}{\partial y \partial x}$ を求めるには，

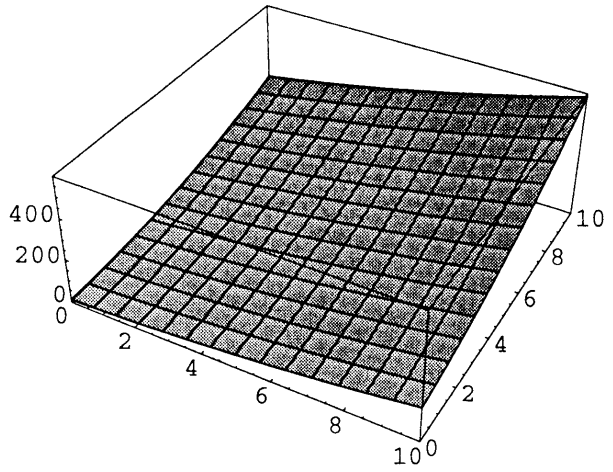
```
In[58]:= D[x^2 + 5*x + 2*y^2 + 3*y + 2*x*y, x, y]

Out[58]= 2
```

とすればよい。

この関数 $x^2+5x+2y^2+3y+2xy$ のグラフを描いてみよう。これは2変数関数であり，グラフは3次元グラフになる。3次元グラフィックスを描くには，組み込み関数 Plot3D を用いる。

```
In[59]:= Plot3D[x^2 + 5*x + 2*y^2 + 3*y + 2*x*y,
                {x, 0, 10}, {y, 0, 10}]
```




```
Out[59]= - SurfaceGraphics -
```

上の3次元グラフは左側に原点があり，x 軸は原点から右下方に10まで，y 軸は右上方に10までのび，縦方向に関数値を測っている。3次元グラフィックスでは，どの視点からグラフを眺めるかを決める空間点を設定することが特に重要になる。視点の設定により，3次元グラフィックスは見易くもなり見難くもなる。視点の設定には，オプション ViewPoint（V と P は大文字）を利用する。

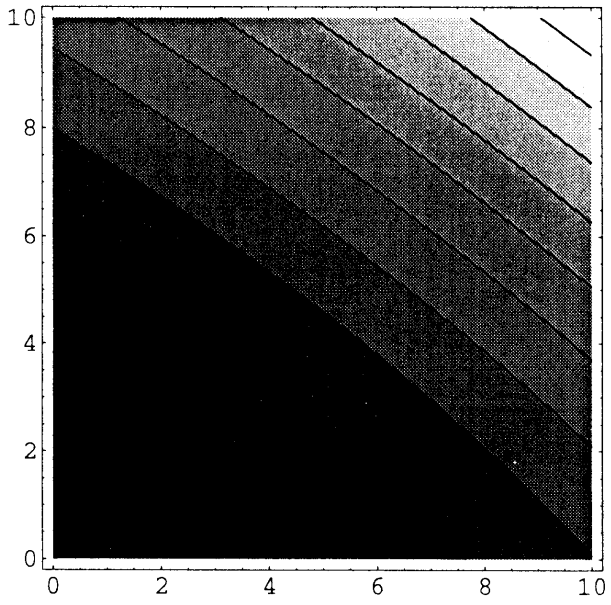
この組み込み関数は

```
ViewPoint->{x, y, z}
```

の形で視点の座標を指定するが，引数 x, y, z の設定が難しいので，「入力」から「3D ビューポイントの設定」を選んで，スクロールバーを操作すると立方体が回転するので，実際に視点を確認しながら引数 x, y, z の値を決定するのが便利である。引数の値が決定したら，ノートブックの入力配列の ViewPoint オプションを設定したい箇所にカーソルを合わせておいて，ペーストをクリックするだけで，入力配列にオプションが記述される。オプション ViewPoint の効果については，次節の効用の丘のグラフ (In[1] と Out[1]) を参照して欲しい。

3次元グラフィックスを平面（2次元空間）に表現する工夫が等高線である（ミクロ経済学で使われる無差別曲線や等産出量曲線は，この例である）。等高線グラフを描くには，組み込み関数 ContourPlot（C と P は大文字）を用いる。

```
In[60]:= ContourPlot[
           x^2 + 5*x + 2*y^2 + 3*y + 2*x*y,
             {x, 0, 10}, {y, 0, 10}]
```




```
Out[60]= - ContourGraphics -
```

Out[60] の等高線図には，濃淡の影が付いている，囲み枠は表示されているが

横軸と縦軸の座標軸は示されていないなど，グラフィックスの仕様について（小さな）不満がある。これらについては，以下のオプション指定を行うことによって，望むようなグラフを表示させることができる。

`ContourShading -> False`	濃淡の影を付けない。
`Frame -> False`	囲み枠を表示しない。
`Axes -> True`	座標軸を表示する。
`AxesOrigin -> {0, 0}`	座標軸を原点で交差させる。
`AxesLabel -> {"x", "y"}`	横軸に x，縦軸に y という名称を付ける。

これらのオプションの効果については，次節の無差別曲線のグラフ（In[2] と Out[2]）を参照して欲しい。

2.10　線形方程式や多項方程式を解くには，組み込み関数 Solve を用いる。例えば，

$$2x-45=0$$

を x について解くには，以下のようにコマンドを表現すればよい。大小関係の等号としては == （等号を2つ）用いなければならないことに注意せよ。

```
In[61]:= Solve[2 x - 45 == 0, x]
Out[61]= {{x → 45/2}}
```

連立方程式も同様に解くことができる。例えば，x と y の連立方程式

$$\begin{cases} 2x-5y=0 \\ xy=2 \end{cases}$$

を解くには以下のようにコマンドを表現すればよい。既に注意したように，ここで xy とは xy という2文字からなる1つの変数ではなく，x と y の積の意味で用いているのでコマンドでもそのように表現しなくてはならない。

また，y を既に関数として使用しているので，その定義を解消しておかなくてはならない。そのためには組み込み関数 Clear を用いる。入力 In[62] の最後は；（セミコロン）で終わっているが，これはその入力に対する出力を抑制する。

```
In[62]:= Clear[y];

In[63]:= Solve[{2 x - 5 y == 0, x * y == 2}, {x, y}]

Out[63]= {{x → -√5, y → -2/√5}, {x → √5, y → 2/√5}}
```

2.11　ここで，関数を定義する際の，同時割り当て＝（等号）と遅延割り当て：＝（コロンと等号）の違いを説明しておこう。両方とも「割り当て」であるが，異なる使われ方をする。サイコロの例を使い，両者の違いを説明しよう。

サイコロは1から6までの整数の中から1つの数字をランダムに選び出すものと考えられるから，Mathematica ではある領域からランダムに数字を選ぶ組み込み関数 Random を使って，

```
Random[Integer, {1, 6}]
```

と表すことができる。

最初に，＝（同時割り当て）により saikoro1（サイコロ1）を定義してみよう。

```
In[64]:= saikoro1 = Random[Integer, {1, 6}]

Out[64]= 6
```

出力配列 Out[64] の6という結果は，saikoro1 を振ったとき（たまたま）6という目が出たことを意味する。これ以降，saikoro1 を評価すると

```
In[65]:= saikoro1

Out[65]= 6
```

と，同じ結果を表示する。すなわち，＝（同時割り当て）では，最初の割り当てが維持される。

次に，：＝（遅延割り当て）により別の saikoro2（サイコロ2）を定義してみよう。

```
In[66]:= saikoro2 := Random[Integer, {1, 6}]
```

遅延割り当ての場合には，この段階では結果を出さない。そして saikoro 2 と入力して評価すると

```
In[67]:= saikoro2
Out[67]= 4
```

と，この段階で結果を出す。さらに，saikoro2 と入力して評価すると

```
In[68]:= saikoro2
Out[68]= 5
```

と，今度は別の結果を出す。以後，saikoro2 と入力して評価するとそれぞれのセッションで新たに割り当てが行われ評価される。すなわち，saikoro2 はサイコロであり，振るたびに新たな目を示す（結果を出す）のに対して，saikoro1 はあるサイコロを1回振ったときの目（結果）であるといえよう。

ある関数の値を調べるにも，=（同時割り当て）と，:=（遅延割り当て）では方法が異なる。例えば，関数 $f(x,y)=x^3y^4$ の $x=2, y=3$ のときの値を知るには，=（同時割り当て）で定義している場合には，置き換え /.（スラッシュとピリオド）を用いる。

```
In[69]:= x^3*y^4
Out[69]= x^3 y^4
In[70]:= % /. {x -> 2, y -> 3}
Out[70]= 648
```

また，関数を :=（遅延割り当て）で定義している場合には，次のようにすればよい。すなわち，x と y に _（アンダースコア）を付けて引数（独立変数）として，関数を

```
In[71]:= f[x_, y_] := x^3*y^4
```

と定義する。この段階では結果は得られない。次に，x に2，y に3という値を与えたときの関数 f の値を評価すると，

```
In[72]:= f[2, 3]
Out[72]= 648
```

という結果が得られる。したがって関数を定義するときは，:=（遅延割り当て）

を用いることが便利であることが多い。

2.12　最後に，統計処理を行うには統計関数のパッケージ Statistics を，複雑なグラフを表示するにはパッケージ Graphics を呼び出して読み込む必要がある。パッケージの読み込みには，<<（不等号を2つ，組み込み関数 Get）を使うが，詳しくは Wolfram (1996, section 1.3.10) を参照して欲しい。

以上で，経済学において利用する Mathematica の基本的な操作法の説明を終わる。次節から，ここで紹介した機能を使いミクロ経済学の基本的な問題を実際に解いてみよう。

3.　消費者行動

3.1　本節では，2種類の財が取り引きされている完全競争市場を想定して，与えられた予算の下で消費から得られる効用の最大化を図る消費者の行動から，この消費者の2財の需要関数を導いてみよう。ただし，効用関数の関数形は Cobb-Douglas 型であるとする。すなわち，この消費者の2財の需要量を x と y とするとき，効用関数が

$$u(x,y) = x^a y^{1-a} \qquad 0 < a < 1 \tag{3.1}$$

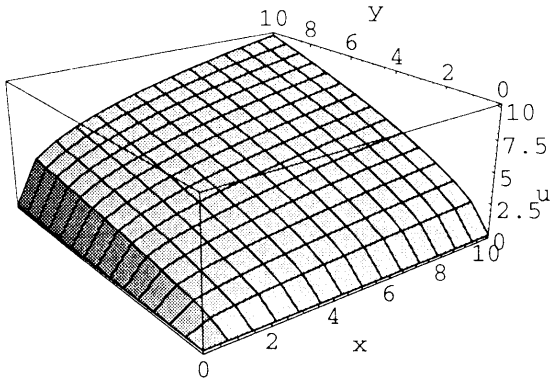
と表される場合を考えよう。

最初に，この効用関数の形状を調べておこう。Mathematica のグラフィックス機能を使うには，この場合は2財の需要量と効用の3次元のグラフになるので，組み込み関数 Plot3D を使用する。a の値を特定しないとグラフは描けないので，ここでは $a = 0.4$ とおこう。すなわち，Cobb-Douglas 型効用関数

$$u(x,y) = x^{0.4} y^{0.6}$$

のいわゆる効用の丘のグラフは，次により描かれる。

```
In[1]:= Plot3D[x^0.4*y^0.6, {x, 0, 10}, {y, 0, 10},
                AxesLabel -> {x, y, u},
                ViewPoint -> {-1.520, -2.010, 0.990}]
```

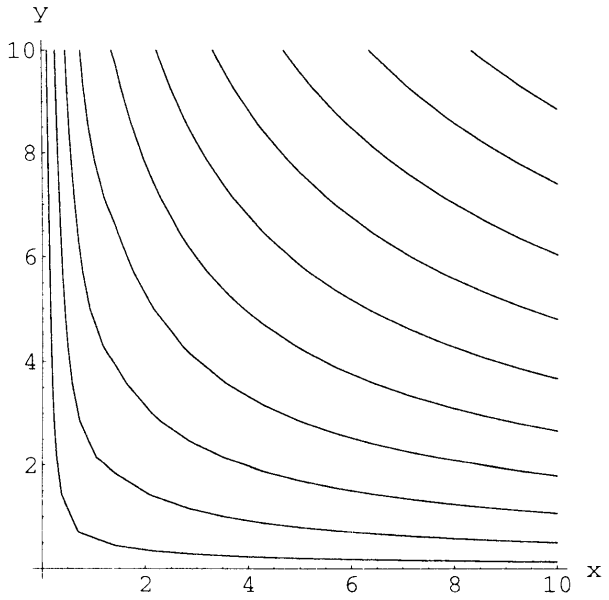



```
Out[1]= - SurfaceGraphics -
```

ここでは，組み込み関数 Plot3D のオプション AxesLabel と ViewPoint を使った。

この効用関数について無差別曲線のグラフを描いてみよう。無差別曲線は，上で描いた効用の丘の等高線に他ならないから，組み込み関数 ContourPlot を使う。上と同様に，$a = 0.4$を想定すると，

```
In[2]:= ContourPlot[x^0.4*y^0.6, {x, 0, 10}, {y, 0, 10},
        ContourShading -> False, Frame -> False,
        Axes -> True, AxesOrigin -> {0, 0}, AxesLabel ->
        {"x", "y"}]
```




```
Out[2]= - ContourGraphics -
```

とすればよい。

アニメーションを使えば，効用関数 (3.1) のパラメーター a が変化するにつれて，無差別曲線の形状がどのように変わるかを連続的に観察できる。Mathematica のアニメーションは，連続したグラフィックス・セルを次々とコマ送りすることによって，グラフィックスを動画化するものである。このためには，組み込み関数 Table を

```
Table［グラフィックスの入力式，
        {パラメーター，初めの値，終わりの値，刻み}］
```

の形で使う。具体的には，上の In[2] の入力式の 0.4 と 0.6 をパラメーター a と $(1-a)$ に戻して，

```
In[3]:= Table[ContourPlot[x^a*y^(1-a),
          {x, 0, 10}, {y, 0, 10},
          ContourShading -> False, Frame -> False,
          Axes -> True, AxesOrigin -> {0, 0}, AxesLabel ->
          {"x", "y"}], {a, 0.1, 0.9, 0.1}]
```

とすれば良い。a が 0.1 から 0.9 まで 0.1 刻みで変化するときの無差別曲線のグラフが出力される（複数，本稿では省略）から，どれか 1 つのグラフをダブルクリックすればアニメーションが始まる。無差別曲線が，a が 0 に近い場合の L 字形から，$a=0.5$ に近付くと直角双曲線になり，a がさらに大きくなって 1 に近付くと再び L 字形に変化していく様子が分かるであろう。

3.2　この消費者の予算は m であるとすると，予算制約の下での効用最大化問題は

$$\max_{x,y} u(x,y) \quad \text{subject to } p_x x + p_y y = m \tag{3.2}$$

と定式化できる。ただし，p_x と p_y は 2 財の市場価格である。この問題を Mathematica を使って解いてみよう。解き方は幾通りかある。

予算制約の下での効用最大化という主体均衡を見付ける最も初歩的な方法は，無差別曲線と予算線の接点を見付けるという図を使った解法であろう。問題 (3.2) の解が無差別曲線と予算線の接点として与えられるという事実から，最適解の満たすべき条件として，(i)最適解においては，無差別曲線の傾き（＝限

界代替率）と予算線の傾き（＝相対価格）が等しい（必要条件）ことと，(ii)最適解は予算線の上にある，すなわち予算制約式が満たされる（十分条件）こととという２つの条件が導かれる。

したがって，制限付き最大化問題 (3.2) の第１の解き方として，限界代替率 MRS＝相対価格の条件

$$\text{MRS} = \frac{p_x}{p_y} \tag{3.3}$$

と予算制約式

$$p_x x + p_y y = m \tag{3.4}$$

を連立させて解くことを検討してみよう。ここで，限界代替率は

$$\text{MRS} \equiv \frac{\partial u}{\partial x} \Big/ \frac{\partial u}{\partial y} \tag{3.5}$$

と定義されるから，(3.3) は

$$\frac{\partial u}{\partial x} \Big/ \frac{\partial u}{\partial y} = \frac{p_x}{p_y} \tag{3.3'}$$

と書き換えられる。

以上の計算を行う Mathematica のノートブックは，次のようになる。

```
In[4]:= u = x^a * y^(1 - a)

Out[4]= x^a y^(1-a)

In[5]:= MRS = D[u, x] / D[u, y];

In[6]:= Solve[{MRS == px / py, px * x + py * y == m}, {x, y}]

Out[6]= {{x → a m / px, y → -((-1 + a) m) / py}}
```

入力配列 In[4] では，Cobb-Douglas 型の効用関数 (3.1) を定義している。ここでは，ベキ乗を表すのに記号 ^（ハット）を使用している。In[5] では，限界代替率 (3.5) を定義している。ここでは，入力の最後に；(セミコロン) を付けて，この入力に対する出力を抑制している。もし；を付けなければ，Mathematica は効用関数が与えられた関数形 (3.1) である場合の限界代替率を求め，$\frac{ay}{(1-a)x}$ という結果を出力する。

In[6] の組み込み関数 Solve は，限界代替率＝相対価格の条件 (3.3') と予算制約式 (3.4) からなる連立方程式体系を 2 財の需要量 x と y を未知数として解く命令である。なお，version 3 以降の Mathematica であれば，基礎的な入力パレットを使って下付き（あるいは上付き）添え字を入力することも可能であるが，ここでは入力の便宜を考えて，価格変数 p_x と p_y を簡単に px と py と表している。出力配列 Out[6] は，連立方程式を解く命令 In[6] に対する結果であり，両財の需要関数が

$$x = \frac{am}{p_x} \tag{3.6}$$

$$y = \frac{(1-a)m}{p_y}$$

という周知の式になることを示している。

3.3　制約付き最大化問題 (3.2) を解くには，Lagrange の未定乗数法を使う方法も考えられる。第 2 の解き方として，これを検討してみよう。

Lagrange の未定乗数法とは，Lagrange 乗数 λ を導入して，制約付き最大化問題 (3.2) の目的関数と制約条件から x, y, λ を変数とする Lagrange 関数

$$L(x,y,\lambda) = u(x,y) + \lambda(m - p_x x - p_y y) \tag{3.7}$$

を構築して，2 変数の制約付き最大化問題 (3.2) を 3 変数の制約のない最大化問題

$$\max_{x,y,\lambda} L(x,y,\lambda) = u(x,y) + \lambda(m - p_x x - p_y y) \tag{3.8}$$

に書き換えた上で，最大化の 1 階の条件から最適解を求める方法である。最適解は最大化の 1 階の条件

$$\frac{\partial L(x,y,\lambda)}{\partial x} = 0 \tag{3.9}$$

$$\frac{\partial L(x,y,\lambda)}{\partial y} = 0$$

$$\frac{\partial L(x,y,\lambda)}{\partial \lambda} = 0$$

を満たすから，この3本の1階の条件を連立させて解き最適解を求めるのである。

この方針で制約のない最大化問題 (3.8) を解き，効用最大化行動から各財の需要関数を求めてみよう。ここで効用関数の具体的な関数形は (3.1) で与えられているから，Lagrange 関数 (3.7) は

$$(3.7') \qquad L(x,y,\lambda) = x^a y^{1-a} + \lambda(m - p_x x - p_y y)$$

と書き換えられる。この Lagrange 関数 (3.7') を Mathematica のノートブックに入力する。

```
In[7]:= L = x^a*y^(1 - a) + lambda (m - px*x - py*y)
Out[7]= x^a y^(1-a) + lambda (m - px x - py y)
```

なお入力配列 In[7] では，ギリシャ文字のをその英語名を使い lambda と表しているが，escape キー，l（小文字のエル），escape キーの順にキーを押すと，ギリシャ文字をノートブックに入力することができる[8]。version 3 以降の Mathematica であれば，基礎的な入力パレットを使うことも可能である。

次に，1階の条件 (3.9) を求める。

```
In[8]:= foc = {D[L, x] == 0, D[L, y] == 0, D[L, lambda] == 0}
Out[8]= {-lambda px + a x^(-1+a) y^(1-a) == 0,
         -lambda py + (1 - a) x^a y^-a == 0, m - px x - py y == 0}
```

そして，1階の条件から最適解を求めようとしても，

```
In[9]:= Solve[foc, {x, y, lambda}]
        Solve::tdep : 方程式に本質的に非代数的な変数の超越関数が含まれている可能性が大きいです.
Out[9]= Solve[{-lambda px + a x^(-1+a) y^(1-a) == 0, -lambda py +
        (1 - a) x^a y^-a == 0, m - px x - py y == 0}, {x, y, lambda}]
```

という警告メッセージが出て，望む結果は得られない[9]。筆算ならば，第2の

8) このように escape キーを使えば，他のギリシャ文字も入力することができる。詳しくは，Wolfram（1996, 訳書 1998, 1.10.1）参照。なお，経済学で利潤を表す変数として使われることが多い π（とその英語名 Pi）は，円周率を表す数学定数として予約されているので注意を要する。

解法でも Cobb-Douglas 型のままで解けるのに，Mathematica で解くには少し工夫が必要である。

そこで，ある関数が効用関数であれば，その関数に順序保存的な変換を施して得られる関数もやはり効用関数として利用できるという事実を利用する。具体的には，Cobb-Douglas 型効用関数 (3.1) の両辺の対数を取り，左辺の $\log u(x, y)$ を改めて $v(x, y)$と置き，右辺を整理して得られる対数線形の関数

$$v(x,y) = a\log x + (1-a)\log y \tag{3.10}$$

を求める。対数変換は単調増加変換であり，したがって順序保存的であるから，このようにして得られた (3.10) も Cobb-Douglas 型と同値な効用関数となる。

この対数線形の効用関数 (3.10) を使って，制約のない最大化問題 (3.7) を解いて需要関数を求めてみよう[10]。この場合の Lagrange 関数

$$L(x,y,\lambda) = a\log x + (1-a)\log y + \lambda(m - p_x x - p_y y) \tag{3.7"}$$

は，Mathematica では

```
In[10]:= L = a*Log[x] + (1 - a)*Log[y] + lambda*(m - px*x - py*y)
Out[10]= lambda (m - px x - py y) + a Log[x] + (1 - a) Log[y]
```

と表される。ただし，対数を表す組み込み関数 Log は，底を指定していないので自然対数である。自然対数の微分は

$$\frac{d\log x}{dx} = \frac{1}{x}$$

となる性質により，以下の計算は容易になる。

1階の条件から最適解を求めると，

```
In[11]:= foc = {D[L, x] == 0, D[L, y] == 0, D[L, lambda] == 0}
Out[11]= {-lambda px + a/x == 0, -lambda py + (1 - a)/y == 0,
          m - px x - py y == 0}
```

9) 山下 (1995) は，「これは Mathematica のプログラムとしての限界ではなく，5次以上の高次方程式を解くときの根本的な数学上のむずかしさに起因する」としている。あわせて，Wolfram (1990)（白水訳（1992, p. 589））参照。

```
In[12]:= Solve[foc, {x, y, lambda}]
```

$$\text{Out[12]= } \left\{\left\{\text{lambda} \to \frac{1}{m},\ x \to \frac{a\,m}{px},\ y \to -\frac{(-1+a)\,m}{py}\right\}\right\}$$

となって，当然のことであるが第1の方法と同じ結果 (3.6) が得られる。さらに，Lagrange 乗数が所得の限界効用と解釈されることも明らかになる。

効用関数としての Cobb-Douglas 型と対数線形の同値性は経済学の知識としては周知の事実であるが，Mathematica にはこの知識が欠けているために，Cobb-Douglas 型のままでは結果が出ないのに，対数線形にすると結果が出るものと思われる[11]。

3.4 最後に，2財の場合すなわち目的関数が2変数関数である場合には，制約条件（予算制約式）(3.4) を使って1変数を消去し，目的関数を残る1変数のみの関数に書き換えて，制約のない1変数関数の最大化問題に変換した上で，最適化の1階の条件から最適解を求める方法もある。これを第3の方法として検討しよう。

ここでは，制約条件 (3.4) を y について解いて

$$y = \frac{1}{p_y}(m - p_x x) \tag{3.11}$$

を求め，これを目的関数（効用関数）(3.1) に代入して y を消去し，

$$u = x^a \left(\frac{m - p_x x}{p_y}\right)^{1-a} \tag{3.1'}$$

と，目的関数を x のみの関数に書き換えて，最大化の1階の条件 $\frac{du}{dx} = 0$ から x を求めるという方針で計算してみよう。

これを Mathematica で行うには，先ず予算制約式 (3.4) を y について解き，

```
In[13]:= Solve[px*x + py*y == m, y]
```

$$\text{Out[13]= } \left\{\left\{y \to -\frac{-m + px\,x}{py}\right\}\right\}$$

これを効用関数 (3.1) に代入して，目的関数から y を消去して x のみの関数と

10) Huand and Crooke (1997, Example 11.25, p. 422)，浅利ほか（1997, 第4章例題 4–1, pp. 95-102）も対数線形の効用関数を想定している。

11) 吹春俊隆先生（広島大学）の指摘。

する。

```
In[14]:= Simplify[x^a*y^(1 - a)] /. %
```

$$\text{Out[14]}= \left\{x^a \left(-\frac{-m + px\,x}{py}\right)^{1-a}\right\}$$

次に，最大化の1階の条件 $\frac{du}{dx} = 0$ を求めるために，上で求めた効用関数を微分して0とおく。

```
In[15]:= D[%, x] == 0
```

$$\text{Out[15]}= \left\{a\,x^{-1+a}\left(-\frac{-m + px\,x}{py}\right)^{1-a} - \frac{(1 - a)\,px\,x^a\,\left(-\frac{-m+px\,x}{py}\right)^{-a}}{py}\right\} == 0$$

そして，この1階の条件を x について解くと，

```
In[16]:= Solve[%, x]

Solve::ifun : 逆関数がSolveにより使用されているので，求められない解のある可能性があります．
```

$$\text{Out[16]}= \left\{\left\{x \to \frac{a\,m}{px}\right\}\right\}$$

と警告メッセージは出るものの，望む結果が得られる。y の需要関数は，この x を (3.11) に代入すれば求められる。

```
In[17]:= Simplify[-(-m + px x)/py] /. %
```

$$\text{Out[17]}= \left\{\frac{m - a\,m}{py}\right\}$$

4. 生産者行動

4.1 本節では，完全競争市場において2種類の生産要素（資本と労働）を投入して1種類の財を産出する生産者を取り上げよう。完全競争を前提にするということは，財価格 p，賃貸率（資本要素価格）r，賃金率（労働要素価格）w，また生産規模に関係なく生じる固定費用 C_0 はこの生産者が決定できるものではなく，パラメーターと見なされることを意味する。

生産者の行動基準として，保有している生産技術の下での利潤の最大化という合理的行動仮説を採用し，供給関数を導いてみよう。この生産者の生産技術

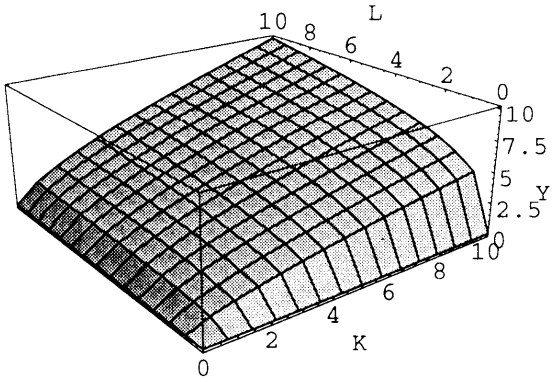
は，Cobb-Douglas 型生産関数

(4.1) $Y = K^b L^{1-b}$ $0 < b < 1$

で与えられるものとする。ただし，Y は財の産出量であり，K は資本の，L は労働の投入量である。なお，ここでは資本 K の係数 b と労働 L の係数 $1-b$ の和が丁度1になる場合を想定している。これは，数学的には生産関数が1次同次であること，経済学的には規模に関する収穫不変の性質を持つことを意味する。

生産関数は，最も効率的に生産を行った場合の要素投入量と産出量の間の技術的な関係を示している。供給関数の導出に取りかかる前に，この生産関数の形状を組み込み関数 Plot3D を使って調べておこう。パラメーター b の値を特定しないとグラフは描けないので，ここでは $b = 0.7$ とする。

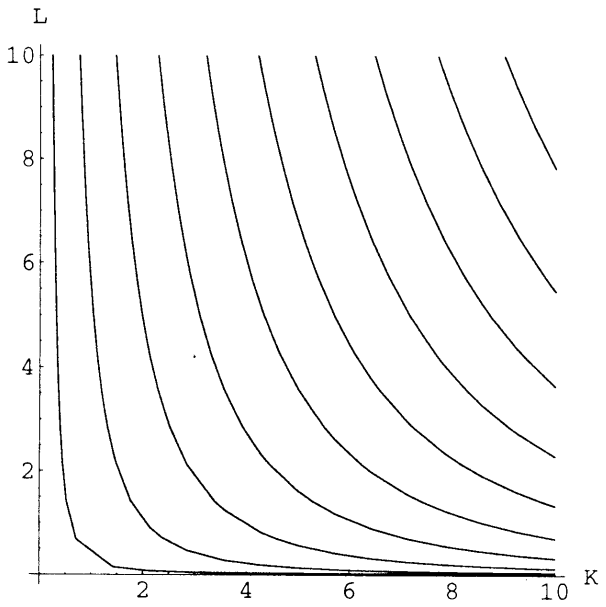
```
In[1]:= Plot3D[K^0.7 * L^0.3, {K, 0, 10}, {L, 0, 10},
          AxesLabel -> {K, L, Y},
          ViewPoint -> {-1.520, -2.010, 0.990}]
```




```
Out[1]= - SurfaceGraphics -
```

等産出量曲線（等量線，isoquant curve）はこの3次元グラフィックスの等高線に他ならないから，等産出量曲線を描くには組み込み関数 ContourPlot を使って次のようにする。

```
In[2]:= ContourPlot[K^0.7*L^0.3,
         {K, 0, 10}, {L, 0, 10},
         ContourShading -> False, Frame -> False,
         Axes -> True, AxesOrigin -> {0, 0}, AxesLabel -> {"K", "L"}]
```




```
Out[2]= - ContourGraphics -
```

ここでも，アニメーション機能を使えば，生産関数のパラメーター b が変化するにつれて，等産出量曲線の形状がどのように変わるかを連続的に観察することができる。

利潤は収入 pY マイナス費用 $rK + wL + C_0$ と定義されるから，上の行動仮設は生産関数 (4.1) を用いると，

$$\max_{Y,K,L} pY - (rK + wL + C_0) \quad \text{subject to } Y = K^b L^{1-b} \tag{4.2}$$

と表される。この制約付き最大化問題 (4.2) を解くことにより，最適な要素投入量（要素需要関数）と産出量（供給関数）が決定されるが，これは通常，(i)生産者の持つ生産技術を費用関数として把握し直し，(ii)費用関数を使って最適産出量を決定するという２段階に分けて考察される。ここで費用関数とは，要素価格が与えられているときに，所与の産出量を最も効率的に生産する場合の産出量と最小の費用の関係を示すものである。

4.2　第1段階の費用関数の導出には，産出量を任意の水準 $\overline{Y}$ に固定しておき，$Y=\overline{Y}$ の追加条件の下で問題 (4.2) を解く。このとき，最大化すべき利潤は $p\overline{Y}-(rK+wL+C_0)$ と表されるが，収入 $p\overline{Y}$ は定数となるので，利潤最大化は費用 $rK+wL+C_0$ を最小にすることと同値である。さらに，固定費用 C_0 は生産規模に関係ないので，費用最小化は可変費用 $rK+wL$ の最小化と同値である。したがって，問題 (4.2) は

$$\min_{K,L} rK+wL \quad \text{subject to } \overline{Y}=K^bL^{1-b} \tag{4.3}$$

を解くことと同値になる。

この最小化問題 (4.3) は，「産出量が $\overline{Y}$ となる生産要素投入 (K, L) の組み合わせの中から，可変費用 $rK+wL$ を最小にするものを見付ける」ことを意味している。ここで，問題 (4.3) の制約条件「産出量が固定された水準となる生産要素投入 (K, L) の組み合わせ」を表すグラフが，等産出量曲線である。一方，可変費用 $rK+wL$ を $\bar{C}$ とおいた方程式のグラフは，可変費用が一定値 $\bar{C}$ となる生産要素投入 (K, L) の組み合わせを示しており，(固定費用を差し引いた) 等費用線と呼ばれる。これは傾きが要素価格比率の直線であり，各費用水準に応じて1本ずつ描かれるが，左下に位置する等費用線ほど対応する費用水準は低くなる。

したがって，最小化問題 (4.3) の解は，固定された産出量水準に対応する等産出量曲線と等費用線の接点として見付けることができる。すなわち，最適解の満たすべき条件は，(i)最適解においては，等産出量曲線の傾き（＝技術的限界代替率）と等費用線の傾き（＝要素価格比率）が等しいこと（必要条件）と，(ii)最適解は産出量水準に対応する等産出量曲線の上にあること（十分条件）の2つである。

そこで，条件(i)の技術的限界代替率 MRS＝要素価格比率

$$MRS=\frac{r}{w} \tag{4.4}$$

と条件(ii)の最適解は等産出量曲線 Y の上にある，すなわち

$$\overline{Y}=K^bL^{1-b} \tag{4.5}$$

を連立させて解くことが考えられる。ただし，技術的限界代替率は

$$\text{MRS} \equiv \frac{\partial F}{\partial K} \Big/ \frac{\partial F}{\partial L} \tag{4.6}$$

と定義されるから，(4.4) は

$$\frac{\partial F}{\partial K} \Big/ \frac{\partial F}{\partial L} = \frac{r}{w} \tag{4.4'}$$

と書き換えられる。

以上の計算を行う Mathematica のノートブックは，次のようになる。先ず，生産関数 (4.1) を定義し，

```
In[3]:= F = K^b * L^(1 - b)
```

$$\text{Out[3]=}\ K^{b}\, L^{1-b}$$

技術的限界代替率 (4.6) を求める。

```
In[4]:= MRS = D[F, K] / D[F, L]
```

$$\text{Out[4]=}\ \frac{b\,L}{(1-b)\,K}$$

技術的限界代替率＝要素価格比率という第１の条件（最適解の必要条件）を労働投入量 L について解き，

```
In[5]:= Solve[MRS == r / w, L]
```

$$\text{Out[5]=}\ \left\{\left\{L \to -\frac{(-1+b)\,K\,r}{b\,w}\right\}\right\}$$

生産関数 (4.1) に代入して，生産関数を資本投入量 K のみで表す。

```
In[6]:= Simplify[F] /. %
```

$$\text{Out[6]=}\ \left\{K^{b}\left(-\frac{(-1+b)\,K\,r}{b\,w}\right)^{1-b}\right\}$$

これらの要素投入量を使えば，固定された産出量を生産することができるという第２の条件（最適解の十分条件）を考慮するために，固定された産出量を生産するのに必要とされる要素投入量を求める。具体的には，上で求めた資本投入量 K のみで表した生産関数の値をにおいて，資本投入量 K について解く。ただし，入力配列 In[7] では，$\bar{Y}$ を簡単に Y と表している。

```
In[7]:= Solve[ (-(-1 + b) * r / (b * w)) ^ (1 - b) * K == Y, K]

Out[7]= {{K → ((1 - b) r / (b w))^(-1+b) Y}}
```

出力配列 Out[7] の結果は，産出量が Y であるときの最適資本投入量（資本需要）$K^d(Y)$ が

$$K^d(Y) = \left(\frac{b}{1-b}\frac{w}{r}\right)^{1-b} Y \tag{4.7}$$

で与えられることを示している。(4.7) より，資本需要は，資本賃貸料（資本の要素価格）r の減少関数，賃金率（労働の要素価格）w の増加関数，産出量 Y の増加関数であることが確認される。これを出力配列 Out[5] の結果に代入して，最適労働投入量 $L^d(Y)$ を求める。

```
In[8]:= Simplify[(-(-1 + b) * ((1 - b) * r / (b * w)) ^ (-1 + b)
          * Y * r / (b * w)) ^ (1 - b)]

Out[8]= (((r - b r) / (b w))^b Y)^(1-b)
```

すなわち，労働需要は

$$L^d(Y) = \left(\frac{1-b}{b}\frac{r}{w}\right)^{b} Y \tag{4.8}$$

で与えられ，賃金率（労働の要素価格）w の減少関数，資本賃貸料（資本の要素価格）r の増加関数，産出量 Y の増加関数であることが確認される。

可変費用 $rK + wL$ は

```
In[9]:= c = r * ((1 - b) * r / (b * w)) ^ (-1 + b) * Y + w * ((r - b * r) /
          (b * w)) ^ b * Y

Out[9]= r ((1 - b) r / (b w))^(-1+b) Y + ((r - b r) / (b w))^b w Y

In[10]:= Simplify[%]

Out[10]= ((r - b r) / (b w))^b w Y / (1 - b)
```

により求められる。費用関数はこれに固定費用 C_0 を加えたものであるから，

$$C(Y) = \left(\frac{1-b}{b}\frac{r}{w}\right)\frac{w}{1-b} Y + C_0 \tag{4.9}$$

により与えられる。以上で，供給関数導出のための第 1 段階を終わる。

4.3 費用関数が求められたので，第2段階の最適産出量の決定を考察しよう。費用関数 (4.9) を使えば，利潤最大化問題 (4.2) は

$$\max_{Y} pY - C(Y) \tag{4.10}$$

と書き換えられる。当初の利潤最大化問題 (4.2) は制約付き最大化問題であったが，費用関数 (4.9) を使って書き換えた問題 (4.10) は制約条件なしの最大化問題になっている。これは，費用関数の背後に生産関数があり，それぞれの産出水準を最小の費用で生産する資本と労働の投入量の組み合わせの存在が保証されているという費用関数の性質による。

問題 (4.10) は制約条件のないしかも1変数関数の最大化問題であるから，目的関数を Y について微分し0とおけば，最大化の1階の条件

$$p = \frac{dC(Y)}{dY} \tag{4.11}$$

が与えられる。ここで，右辺の費用関数の産出量 Y に関する微分は，限界費用 MC に他ならないから，(4.11) は

$$p = MC(Y) \tag{4.11'}$$

という周知の式に書き換えられる。これが供給関数である。

限界費用を Mathematica で求めよう。固定費用は定数であるから微分すると0になるので，費用関数 (4.9) 全体を微分しなくとも，可変費用（出力配列 Out[10] の結果）を微分すれば十分である。よって，限界費用 MC は

```
In[11]:= MC = D[%, Y]

Out[11]=  ( r - b r / b w )^b w / (1 - b)
```

最後に，これを財価格 p に等しくおけば，供給関数が求められる。

$$p = \left(\frac{1-b}{b}\frac{r}{w}\right)^{b}\frac{w}{1-b} \tag{4.12}$$

(4.12) の右辺は，要素価格 r と w，生産関数のパラメーター b からなる。完全競争の仮定の下では，これらは生産者にとっては与件となるので，(4.12) の右辺は定数となる。したがって，供給関数 (4.12) のグラフすなわち供給曲線は，右辺の値を高さとする水平な直線になる。これは，私たちが規模に関する

収穫不変（1次同次）の生産関数を想定して分析を始めたことの当然の帰結である。

5. むすび

私たちは，完全競争市場における効用最大化行動からの需要関数の導出，利潤最大化行動からの供給関数の導出というミクロ経済学の基本的な問題の検討を通じて，数式処理ソフトウェア Mathematica の経済学における使い方を検討した。Mathematica の有用性が証明されたと同時に，経済学の研究に必要な組み込み関数はそれほど多くないことも明らかになったと思う。他の数式処理ソフトウェアと比較すると，文字式の処理能力に関しては，経済学の分析においては例えば REDUCE 程度で十分であると思われるが，グラフィックス機能に関しては，Mathematica ほど使い易くしかも強力であるソフトウェアは他にはないと考えられる。とりわけ3次元グラフィックス機能やアニメーションは，経済学の基礎概念を理解するのに大変有用である。

ただ，Mathematica とても自動問題解決システムではない。本稿では効用関数や生産関数の関数形として操作が比較的容易な Cobb-Douglas 型を使用したが，それでも解法の方針を工夫しなければならない場面がいくつかあった。CES 型やその他のより複雑な関数形を利用する場合はなおさら，経済学の理論構造を理解することや，どのようなモデルを構築しどのように解くかという工夫が要求されよう。

〔参　考　文　献〕

Huand, C. J., and P. S. Crooke (1997), *Mathematics and Mathematica for Economists,* Blackwell.

Wolfram, S., (1991), *Mathematica : A System for Doing Mathematics by Computer,* 2nd edition,（白水重明訳『*Mathematica : A System for Doing Mathematics by Computer*（日本語版），Second edition』，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，1992）

Wolfram, S., (1996), *Mathematica Book,* 3rd edition, Wolfram Media/Cambridge University Press.（榊原進，武沢護ほか訳『Mathematica ブック（改訂第3版）』，トッパン，1998）

Wolfram, S., (1998), *Mathematica Book,* 3rd edition, Addendum（『Mathematica ブック日本語版第3版追加項目集』，トッパン，1999）

浅利一郎，久保徳次郎，石橋太郎，山下隆文 (1997)『はじめよう経済学のための Mathematica，パソコンによる数式処理』（日本評論社）

小池慎一 (1990)『Mathematica 数式処理入門』(技術評論社)
小平裕 (1985)「経済分析と REDUCE」, 成城大学『経済研究』第98・99合併号
小峯龍男 (1999)『ファーストステップ Mathematica, 数値計算からハイパーリンクまで』(東京電気大学出版局)
山下隆之 (1995)「消費者行動の理論—制約条件付き最適化論 (Mathematica で経済学, 第3回)」『経済セミナー』第487号。
吉田賢史 (2000)『かんたん Mathematica 活用ガイド』(東京電気大学出版局)

Mathematica によるミクロ経済学　　（研究報告　No. 34）

平成14年4月20日　印　刷
平成14年4月25日　発　行　　非売品

著　者　小　平　裕

発行所　成城大学経済研究所
〒157-8511　東京都世田谷区成城 6-1-20
電　話 03（3482）1181番

印刷所　白陽舎印刷工業株式会社