第 2 回：2 期間モデルと数値計算の概観

1　何がどこで必要なのかを理解しよう

連載第 2 回ではシンプルな 2 期間モデルを用いて、数値計算手法の基本的な使い方を説明していく。実際にモデルを解くときに、利用できる分析ツールが自分の道具箱にいっぱい詰まっていた方が選択の幅が広がる。そのため、同じモデルを解くための方法をいくつか紹介しながら、同時に数値計算的発想や特有の専門用語について解説を加える。

例えば、経済学における数値計算の教科書として代表的な Judd (1998) では、第 4 章で**最適化 (optimization)** について説明して、第 5 章で**非線形方程式 (nonlinear equations)** の解法、第 6 章で関数の**近似法 (approximation methods)**、第 7 章で**数値積分と数値微分 (numerical integral and differentiation)** を解説するという手順で進んでいく。本連載を読み通した後であれば、こういった手法が実際に数値計算を行ううえで重要であり、必要に応じてしっかりと理解しなければいけないということにも納得できるはずである。

しかし、どれがいつ役立つのかを知らないまま、ただ数値計算分野でよく使われる手法だからとその数学的性質を学習していっても、ほとんどの人は勉強のモチベーションが保てないのではないか。今回は、実際に簡単な経済モデルを解きながら、各ツールに関する知識が「どこで、なぜ必要になってくるのか」について解説を加えていく。言い換えると、連載第 2 回目は経済学における数値計算の概観をつかむことを目的としている。

2　ベンチマークモデルとカリブレーション

2.1　2 期間モデル

ある経済主体の人生全体での消費・貯蓄行動をモデル化してみよう。経済主体の人生を 2 期間に分けて、前半を若年期 (第 1 期)、後半を老年期 (第 2 期) と呼ぶことにする。彼/彼女は若年期には働いて所得 w を稼得する。所得を今期の消費 c_1 にあてるか、老後のための貯蓄 a として取っておくかに関する意思決定問題を考える。そのため、彼/彼女の若年期の予算制約は

$$c_1 + a = w \tag{1}$$

となる。

老年期になると働くことは出来ないため、若年期に蓄積した資産に頼って生活をする。若年期に

貯蓄した資産には r の金利がつくため、老年期の期初に資産を $(1+r)a$ だけ保有していることになる。第 2 期に経済主体は必ず死亡し、遺産動機はないとする[1]。そうすると、老年期には持っている全財産を消費に回すことが最適になるため、老年期の予算制約は

$$c_2 = (1+r)a \tag{2}$$

となる。

経済主体は人生全体の効用を最大化するように消費と貯蓄を選択する。すなわち、彼/彼女の目的は、予算制約 (1) 式、(2) 式の下で生涯全体の効用水準、

$$U(c_1, c_2) = u(c_1) + \beta u(c_2)$$

を最大にする消費・貯蓄を行うことである。ただし、将来の消費から得られる効用については β だけ割り引くものとする。β は**割引因子 (discount factor)** と呼ばれている。なお、割引と呼んでいるが、有限期間であれば割引因子が 1 を超えていても問題はない。

効用関数が $u'(c) > 0$ 及び $u''(c) < 0$ を満たす場合、経済主体は若年期と老年期で極端に消費水準が変動することを嫌い、**消費の平準化 (consumption smoothing)** を望む。その結果、経済主体は若年期の消費から得られる限界効用 $u'(c_1)$ と、若年期の貯蓄による老年期の追加的消費から得られる限界効用の割引現在価値 $(1+r)\beta u'(c_2)$ が一致するように、消費を決定する。

$$u'(c_1) = (1+r)\beta u'(c_2) \tag{3}$$

(3) 式は**オイラー方程式 (Euler equation)** と呼ばれている。

さて、このシンプルな 2 期間モデルをどうやって解けば良いのであろうか。実際に数値計算手法を説明する前に、何を持ってこのモデルを「解いた」というべきなのかについて、確認しておこう。われわれが知りたいのは、この経済主体が若年期と老年期にどれだけ消費をして、どれだけ貯蓄をするかである。すなわち、ある所得 w のもとで個人の貯蓄関数

$$a = g(w)$$

を導出したい。

[1] 子どもの効用水準が自身の効用に影響を与える場合、遺産動機が発生する。そのようなモデルは王朝 (dynasty) モデルと呼ばれ、無限期間生きる代表的個人モデルの意思決定と一致することが、Barro (1974) などによってよく知られている。

2.2　カリブレーション

2 期間モデルは学部生レベルのマクロ経済学の教科書にも載っている基本的なモデルであるが、このままでは数値計算をすることは出来ない[2]。というのも、効用関数 $u(c)$ という表現は抽象的すぎて、コンピュータには理解できないからである。そこで効用関数を「特定化」する必要がある。効用関数がどのような形状をしているかは消費者行動を理解するうえで大事なトピックであり、場合によっては政策の効果を左右するほど重要であるが、ここではその点に深入りをすることはせず、マクロ経済学で頻繁に使われる**相対的危険回避度一定 (CRRA：constant relative risk aversion)** 型効用関数を仮定する。

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma}$$

γ は**異時点間の代替の弾力性 (intertemporal elasticity of substitution)** の逆数である。

関数型の特定化に加えてパラメータの選択 (ここでは γ の値) にも同様に慎重になる必要がある。関数形を特定化してパラメータを定める一連の作業のことを、**カリブレーション (calibration)** と呼んでいる[3]。

カリブレーションとは何かについて厳密に定義付けをするのは実は難しい。単純にモデルのパラメータを「適切」に設定することを全てカリブレーションと呼ぶ人もいれば、よりデータに即した推計に近いアプローチのみをカリブレーションと呼ぶ人もいる。今回は実証的な厳密さは重視していないため、使われているモデルは数値例の域を出ない。しかし、経済学的に全く意味のないパラメータを議論しても生産的ではないので、以下では先行研究に倣った値を使用する。

それでは実際にどのようにパラメータを設定すればよいのかを考えていこう。ベンチマークモデルでは人生を 2 期間に分けているので、モデル上の 1 期間は 30 年と想定する。そのため、割引因子 β と金利 r は年率ではなく 30 年間の値を使う。今回は実際に 1 期間を 30 年でカリブレートしている Song et al. (2012) に従うとしよう。割引因子は年率で $\beta = 0.985$ として、1 期間は 30 年なのでそれを 30 乗する ($\beta = 0.985^{30}$)。金利は年率で 2.5% と設定して、同様にモデルの 1 期間にあわせて 30 乗する ($1 + r = 1.025^{30}$)。異時点間の代替の弾力性は議論がわかれる場合もあるが、こちらもよく使われる $\gamma = 2$ としておく。

---

[2] 2 期間モデルを用いたマクロ経済学の理解については、例えば、バロー (2010) の 7 章やウィリアムソン (2009) の第 6 章を参照せよ。

[3] カリブレーションとは何か、どうあるべきかという議論に関心がある読者は Cooley and Prescott (1995) や Kydland and Prescott (1996)、Hansen and Heckman (1996)、Browning et al. (1999) を参照せよ。

2.3　解析的解の性質

数値計算に入る前に、2 期間モデルの**解析的解 (closed-form solution)** の性質を簡単に確認しておこう。今回のモデルくらいシンプルであれば貯蓄関数 $a = g(w)$ を手計算で導出することも可能である。

$$a = \frac{w}{1 + (1+r)\{\beta(1+r)\}^{-1/\gamma}} \tag{4}$$

(4) 式が我々が本来知りたい真の貯蓄関数であり、その形状は若年期の所得 w に対して線形の増加関数であることが読み取れる。

3　離散近似とグリッド

前置きが長くなったが、いよいよ数値計算にチャレンジしてみよう。経済数学を勉強した人であれば**連続 (continuum)** という概念に馴染みがあるはずである。われわれが真に知りたいのは貯蓄関数 $a = g(w)$ であり、所得 w も資産 a も連続的に変化するはずである。しかし、コンピュータは連続という概念をそのままの形では理解できない。そのため、数値計算においては、基本的に変数を I 個の点に**離散化 (discretize)** して考える必要がある。

3.1　グリッド上で計算する

若年期の所得 w が取り得る値は、$w_i \in \{w_1, \dots, w_I\}$ の範囲にあるとしよう。この点の集まりを**グリッド (grid)** あるいは**ノード (node)** と呼ぶ。また、それぞれの点は**グリッドポイント (grid point)** あるいは**評価点 (evaluation point)** と呼ばれている。

グリッドをどうやって取ると精度や計算速度の面で望ましいのかという点は、若干テクニカルになる[4]。そのため、今回は単純に若年期の所得 w は $[0.1, 1]$ 区間の間に 0.1 刻みで 10 個の点

$$w_i \in \{0.1, 0.2, 0.3, ..., 1.0\}$$

として存在していると考えることにしよう。サブスクリプトの i はグリッドの何番目の要素かを表している。$w_i = 0$ を含めると若年期も老年期も消費が出来なくなってしまうため、最小値は必ず正の値である点に注意して欲しい。

[4] 例えば、Maliar et al. (2010) は借り入れ制約近辺に多めのグリッドポイントを配置する方法を提案している。

3.2　状態変数と操作変数が共に離散の場合

まずは、w と a がともに離散的な場合に 2 期間モデルを解く方法をみていこう。資産 a についても離散化をする。ただし、w_i と同じ $\{0.1, 0.2, 0.3, ..., 1.0\}$ としてしまうと、$w = 0.1$ で消費がゼロか負値しか取れないので最適貯蓄が存在しなくなるため、$a_j \in \{0.025, 0.05, 0.075, 0.1, ..., 1.0\}$ と 0.025 刻みで 40 個のグリッドを取ることにする。

経済主体の生涯にわたる効用最大化問題は、各 w_i について、

$$\max_{a \in \{a_1,...,a_J\}} \frac{[w_i - a]^{1-\gamma}}{1-\gamma} + \beta \frac{[(1+r)a]^{1-\gamma}}{1-\gamma} \tag{5}$$

と書き直すことが出来る。w_i と a_j 以外のパラメータについては、すでに値が定まっているので、w が与えられれば、その下での最適な貯蓄額 a も定まる。

組み合わせの可能性は、所得 w と資産 a でそれぞれ 10 個と 40 個なので、400 通りになる。我々が知りたいのは経済主体の効用最大化問題の解なので、それぞれの w_i 毎に、40 種類の a_j の組み合わせでどれが生涯効用を最大にするのかを「総当り」で計算してみればよい[5]。力技のように感じるかもしれないが、わずか 40 通りであれば、全てのケースをとりあえず計算してみて、その中で効用が最大になる値はどれかを目で確認するのも不可能ではない。とはいえ、実際に手を動かすのはやはり大変である。幸いコンピュータは総当りや最大値がどこにあるかを探すといった単純作業は人間より得意であり、(5) 式を全ての組み合わせで計算した上で、例えば MATLAB であれば `max` 関数を用いて最大値の値と場所を探せばよい。

なお、次回紹介予定の**動的計画法 (dynamic programming)** の言葉を使うと、既に決まっている若年期の所得 w を**状態変数 (state variable)** と呼び、意思決定において選択する a を**操作変数 (control variable)** と呼ぶ。また、意思決定関数 (ここでは貯蓄関数) を**政策関数 (policy function)** と呼ぶ[6]。

図 1(a) は、横軸に若年期の貯蓄 (=老年期の資産) を、縦軸に生涯効用 $u(c_1) + \beta u(c_2)$ の値をとったものである。若年期の所得 w_i がそれぞれ 0.5、0.8、1 の場合に、老年期のための貯蓄として 0.025 から 1 までの値を仮に取ったとしたら生涯効用はいくらになるのかを全ての組み合わせで

---

[5] 若年期の資産が少なく、老年期の資産が多いという組み合わせの場合、若年期の消費 $c_1 = w_i - a_j$ が負値になる可能性がある。負の消費量は通常、経済学において定義されず、対数効用関数の場合には負値の対数なので計算も出来ない。そういう可能性を排除するために、理論上取り得ない値については大きな負の効用を仮に与えることによって、結果的に選択されないようにしておくといった工夫が必要である。

[6] 計量経済学で用いる操作変数法の操作変数 (instrument variable) とは異なるため注意してほしい。同様に、政策関数という名称は経済政策と混乱しやすいが、財政・金融政策などの政府の政策とは必ずしも限らない。

計算している。例えば、若年期に資産 w が 0.5 の経済主体が老後のための貯蓄として $a = 0.1$ を選択した場合、生涯効用を計算すると約 -5.53 となる[7]。貯蓄を増やして $a = 0.2$ とした場合、彼/彼女の生涯効用はおよそ -4.85 へと上昇する。生涯効用を最大にする貯蓄は山型のピークである $a = 0.175$ で、そのときの生涯効用は約 -4.8 である。

若年期の資産 w_i ごとに最適な貯蓄 a_j を計算していってつなげた組み合わせが、図 1(b) である。取りうる値がわずか 40 種類しかないため若干ガタガタしているが、これが 2 期間モデルにおける貯蓄関数 $a = g(w_i)$ である。右上がりになっているため、若年期の資産額が多い人ほど、老年期に残す貯蓄額も増えるという基本的な性質が読み取れる。

状態変数と操作変数をともに離散化して、取りうる組み合わせ全てを計算してみるというアプローチは原始的で単純過ぎるように感じるかもしれないが、実は複雑な非線形モデルにも使えるため侮れない。例えば、政策関数の形状や性質すらよくわかっていないモデルを解く場合、とりあえず離散化をして、モデルの性質を大雑把に把握してみる事から理解を深めていくことも可能である。

ただし、このアプローチは計算時間がかかる。状態変数の数が 2 種類になると、最適な貯蓄を計算しなければいけないグリッドポイントの数は 2 倍ではなく 2 乗 (例えば、$10^2 = 100$ 通り) になる。3 種類の状態変数となれば、$10^3 = 1000$ 通りとなってしまう。この性質を**次元の呪い (curse of dimensionality)** と呼ぶ。次元の呪いをいかにして緩和するかについては様々な工夫が提案されてきたものの、基本的な性質として状態変数を増やせば増やすほど計算時間は指数的に増えていく事は理解しておいて欲しい。そのため、いかにして現実的な計算時間で複雑なモデルを高い精度で解くかについて工夫が求められる。

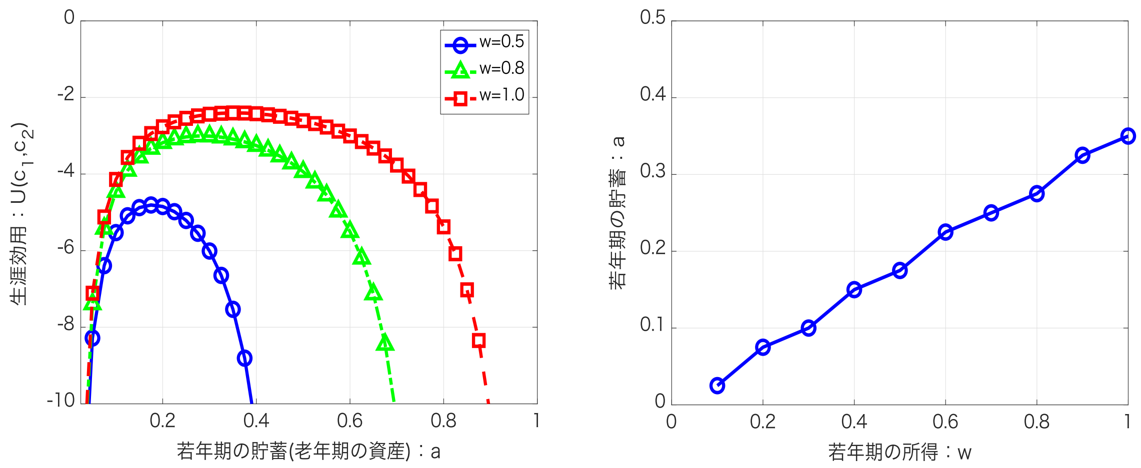
MATLAB Code

- `main_discretization.m`

4 操作変数を連続変数にする：最適化

状態変数と操作変数をともに離散化して考えるアプローチは、モデルを拡張した場合に計算量が指数的に増加する上、離散化するグリッドの数を節約すると精度が非常に悪くなるという問題を抱えている。実際、本来の解である (4) 式は線形であるはずなのに、貯蓄関数を描いた図 1(b) は直線

[7] 効用の値そのものに意味を与える立場を基数的効用、大小関係のみに注目する立場を序数的効用と呼ぶ。数値計算では実際に効用関数に数値を当てはめるため、効用の値も計算していることになるが、数値そのものに重要な意味を見出しているわけではない点に注意してほしい。また、効用の値は関数型に応じて正値・負値のどちらも取り得る。



(a) 生涯効用

(b) 貯蓄関数：$a = g(w)$

図 1: 離散化による 2 期間モデルの解法

ではない。そこで、もう少し洗練されたアプローチを考える。状態変数である w についてはこれまで同様に離散化して考えるが、操作変数 a については連続的な値を取ることを認めよう。

前節と同様に、経済主体の最適化問題は、

$$\max_a \frac{[w_i - a]^{1-\gamma}}{1-\gamma} + \beta \frac{[(1+r)a]^{1-\gamma}}{1-\gamma} \tag{6}$$

となるが、今度は操作変数 $a \in \mathbb{R}$ については任意の実数を取りうるとする。

では、どうやって (6) 式の解を得ればよいのであろうか？通常、数値計算ソフトには**最適化 (optimization)** と呼ばれるライブラリ (あるいは関数、サブルーティン) が存在している[8]。本来、何が実行されているのかまったく理解しないブラックボックスのまま数値計算を行うのは望ましくないし、本連載の趣旨にも反するが、まずは使ってみよう。例えば MATLAB であれば、Optimization Toolbox にある `fminsearch` や `fminbnd` という関数がそれに該当する。また、Python の場合、`scipy` というライブラリに含まれている `fmin` が同様の計算を行う。以下が具体的なアルゴリズムになる。

■アルゴリズム

1. パラメータを設定する (カリブレーション)。
2. $w_i \in \{w_1, ..., w_I\}$ を離散化した若年期の所得とする。

[8] Mablab のように始めからパッケージとして含まれている場合もあるし、追加が必要になる場合もある。また、専門家が書いた最適化アルゴリズムをネットからダウンロードする事も可能である。

3. 各 w_i について、(6) 式を最大にするような a を探し出す。最大値を探すためには、各言語に備わっている (あるいは外部の) 最適化関数を利用する。
4. 得られた各 w_i と a の組み合わせが貯蓄関数である。

MATLAB 関数

- `fminbnd`：黄金分割探索（`Golden section search`）によって極値 (最小値) を探す。
- `fminsearch`：ネルダー-ミード法（`Nelder-Mead method`）によって極値 (最小値) を探す。

図 2 は実際に MATLAB の最適化関数 (`fminsearch`) を使って、各 w_i における最適な a を探してプロットした貯蓄関数である。図 1 と比較してほしい。どちらも横軸に若年期の所得を取り、縦軸に若年期の貯蓄を取った貯蓄関数であるが、ガタガタしていた図 1(b) と違って図 2 はきれいな形をしている。最適化を用いたアプローチも、若年期における資産 w はわずか 10 個の点しか計算していないが、操作変数が取りうる値を自由にする事によって、計算精度が大幅に改善された。

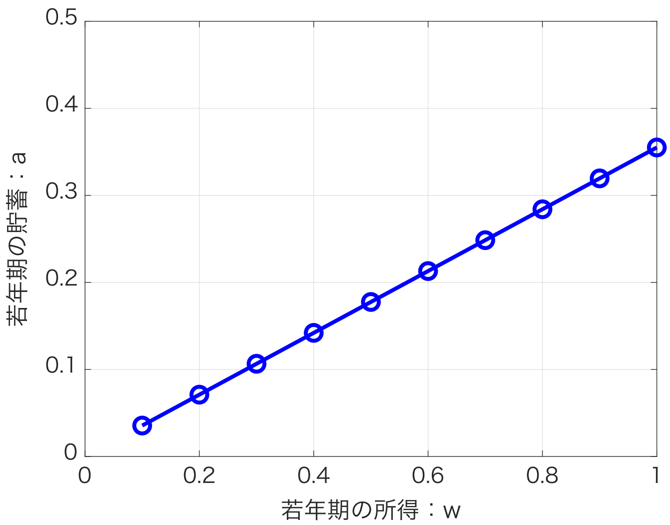


図 2: 最適化から導出した貯蓄関数

MATLAB Code

- `main_optimization.m`、`obj_two_period.m`

5　一階条件を使う：求根アルゴリズム

5.1　非線形方程式のゼロ点を探す

ミクロ経済学をしっかり身につけた読者であれば、おなじみの一階条件 (3) 式を使って計算をしたくなる人もいるかもしれない。予算制約を代入した後の 2 期間モデルの一階条件は、

$$u'(w-a) = \beta(1+r)u'((1+r)a), \tag{7}$$

となる。既に述べたように、マクロ経済学では異時点間の一階条件式をオイラー方程式と呼んでいる。

前節と同様に、状態変数である w_i については離散化して考えよう。そうすると、

$$u'(\underbrace{w_i}_{\text{所与}} - \underbrace{a}_{\text{選択}}) = \underbrace{\beta(1+r)}_{\text{パラメータ}}u'(\underbrace{(1+r)}_{\text{パラメータ}}\underbrace{a}_{\text{選択}})$$

なので、結局のところ未知の変数は a のみである。(7) 式を、

$$R(w_i) \equiv \beta(1+r)\frac{u'((1+r)a)}{u'(w_i-a)} - 1 \tag{8}$$

と書き換えよう。この変換によって、一階条件を解くという事は、ある w_i のもとで**残差関数 (residual function)** $R(w_i)$ の**ゼロ点を探す問題 (root-finding problem：求根問題)** に読み替えることが出来る。一般的に (8) 式のようなオイラー方程式を変換して得られた残差関数は複雑な形をした非線形方程式の可能性があり、ゼロ点を探すことは容易ではない。しかし、幸い非線形方程式のゼロ点を探す手法に関する研究は長い歴史を持つため、既に様々なアプローチが存在している。MATLAB では `fzero`、Python であれば `scipy` ライブラリに含まれている `fsolve` を用いる。

図 3 は、w_i がそれぞれ $\{0.5, 0.8, 1\}$ のときに、若年期の貯蓄 a の値を変化させていったときの残差 $R(w_i)$ の値をプロットしたものである。(8) 式が比較的シンプルなおかげで、残差関数は a に対して単調に減少する関数となっていることがわかる。例えば、図 3 によると $w = 0.5$ の場合には、a が 0.17 近辺でゼロを通過する (=オイラー方程式が等式で成立する) を満たす。これは図 2 で得た貯蓄関数における $w = 0.5$ における最適貯蓄と整合的である。得られた貯蓄関数の図は、図 2 と見た目がまったく変わらないため、割愛する。

■アルゴリズム

1. パラメータを設定する (カリブレーション)。
2. $w_i \in \{w_1, ..., w_I\}$ を離散化した若年期の資産とする。
3. 各 w_i について、(8) 式がゼロになる a を探し出す。ゼロ点を探すためには、各言語に備わっている (あるいは外部ライブラリの) **求根アルゴリズム**を利用する。
4. 得られた各 w_i と a の組み合わせが貯蓄関数である。

MATLAB 関数

- `fzero`：二分法 **(bisection method)**、割線法 **(secant method)** および逆 **2** 次内挿法 **(inverse quadratic interpolation method)** を組み合わせてゼロ点を探す[a]。

[a] 詳細は MATLAB ヘルプを参照。

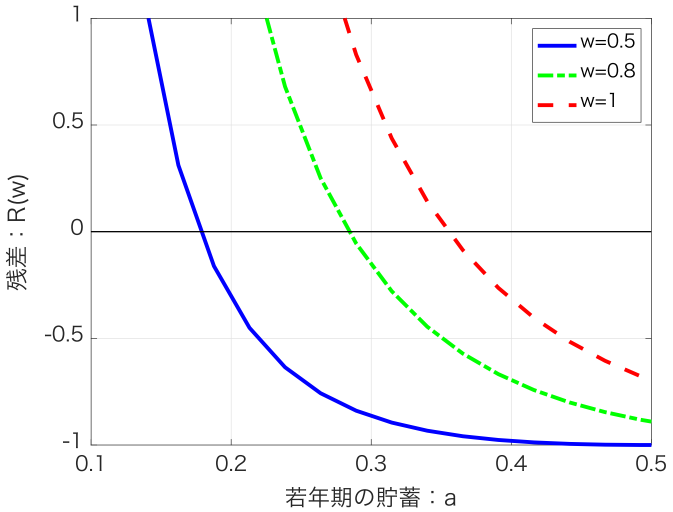


図 3: $w \in \{0.5, 0.8, 1\}$ のときの残差

MATLAB Code

- `main_root_finding.m`、`resid_two_period.m`

5.2　射影法

最適化と求根アルゴリズムを使った上記の手法は、現在の資産水準 w を離散個に区切って、そのうえでの最適貯蓄を計算するという点で共通している。それに対して**射影法 (projection method)** では、政策関数全体を近似する[9]。

貯蓄関数を近似する方法はいくつか考えられるが、例えば、貯蓄関数を N 次の**多項式 (polynomials)**

$$\hat{g}(w;\boldsymbol{\theta}) = \sum_{n=0}^{N} \theta_n w^n \tag{9}$$

で近似したいとしよう。実は今回用いている 2 期間モデルの場合、(4) 式から明らかなように貯蓄関数は線形関数なので、必ず $a = \theta_1 w$ の形式で表現できる。

しかし、一般的には政策関数はより複雑な形状をしている。射影法とは、オイラー方程式を満たす政策関数を探すという問題を、多項式における未知の係数ベクトル $\boldsymbol{\theta} \equiv \{\theta_n\}_{n=0}^{N}$ を探す問題に置き換えるアプローチである。今回はシンプルな多項式を用いているが、より一般的には $\hat{g}(w;\boldsymbol{\theta}) = \sum_{n=0}^{N} \theta_n \Psi_n(w)$ と書くことが出来る。このとき、$\Psi_n(w)$ は**基底関数 (basis function)** と呼ばれる。実は、基底関数の選択によって射影法のアルゴリズムも若干変わってくる[10]。また、次数 N を増やせばそれだけ精度が高くなると考えられるが、計算しなければいけない未知数も増えるため計算が困難になる。どのような場合にも当てはまる適切な N というのは存在しないが、まずは低い次数から試していき、当てはまりが悪ければ次数を増やしていくのが効率的であろう。

ここで前節の残差関数 (8) 式を思いだそう。(9) 式が真の政策関数をうまく近似しているのであれば、この式を残差関数に代入した場合の残差はゼロに近いはずである。すなわち、

$$R(w;\boldsymbol{\theta}) \equiv \beta(1+r)\frac{u'((1+r)\hat{g}(w;\boldsymbol{\theta}))}{u'(w-\hat{g}(w;\boldsymbol{\theta}))} - 1 = 0$$

が「あらゆる w」で成立している必要がある。とはいっても、本来は複雑な非線形関数を多項式で近似しているので、w の定義域上のあらゆる値で残差がちょうどゼロになることは考えられない。十分にゼロに近ければ、近似的にオイラー方程式を満たしているとみなして良いが、問題は何を持ってゼロに近いというべきなのかである。例えば、ある $\boldsymbol{\theta_1}$ を用いて残差を計算した場合には全体的に小さな残差が存在しているに対して、別の $\boldsymbol{\theta_2}$ を用いると所々残差がゼロになるが当ては

[9] 射影法は、**重み付き残差法 (weighted residual method)** とも呼ばれる

[10] 基底関数の中でも頻繁に用いられる**チェビシェフ多項式 (Chebyshev polynomials)** については、連載第 4 回目で紹介する予定である。

まりが悪い箇所もある場合、どちらのほうが良い近似なのだろうか?

当てはまりの良さを定義するために、**距離関数 (metric function)** ρ を導入しよう[11]。一般的に政策関数を探す問題は、

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta} \rho(R(w; \boldsymbol{\theta}), \mathbf{0}) \tag{10}$$

を満たす $\hat{\boldsymbol{\theta}}$ を探す問題に置き換えることが出来る。

(10) 式の距離の取り方は様々であるが、ここではシンプルに任意に評価点 $\{w_i\}$ を取って、そこでの残差がゼロに近くなる係数ベクトル $\boldsymbol{\theta}$ を探すことにしよう。評価点上でのみ距離を測るというのも立派な距離関数であり、これを**選点法 (collocation method)** と呼ぶ。

■アルゴリズム

1. パラメータを設定する (カリブレーション)。
2. 選点 (collocation) $w_i \in \{w_1, ..., w_I\}$ を定める。今回は、これまで使ってきた等分のグリッドと同じで 0.1 から 1 の区間を 0.1 刻みで 10 個設定する。
3. 近似したい政策関数の関数形を決める。今回は 1 次関数 ($\hat{g}(w; \boldsymbol{\theta}) = \theta_0 + \theta_1 w$) とする。
4. ある $\{\theta_0, \theta_1\}$ をインプットとして、選点上の残差 $R(w; \boldsymbol{\theta})$ を計算して返すサブルーチンを書く (`resid_projection.m`)。
5. MATLAB などに備わっているゼロ点を探す関数を使って、残差関数が十分にゼロに近くなる $\{\theta_0^*, \theta_1^*\}$ を見つける。
6. 得られた $\hat{g}(w; \boldsymbol{\theta}^*) = \theta_0^* + \theta_1^* w$ が貯蓄関数である。

実際に数値計算を行った結果を示そう。前述の通り、もともと真の解は線形関数なので、我々が解くべき問題は (10) 式を満たす $a = \theta_0 + \theta_1 w$ の係数を探すという非線形方程式の解となる。傾きは $\frac{1}{1+\{\beta(1+r)\}^{-1/\gamma}(1+r)}$、切片はゼロなので、カリブレーションした値のもとでの真の値は $\theta_0 = 0$ と $\theta_1 = 0.3550$ である。それに対して、数値計算結果である θ_0 と θ_1 はそれぞれ 6.405e-15 と 0.3550 となり、非常に近い値を得ることが出来た。

MATLAB Code

- `main_projection_method.m`、`resid_projection.m`

[11] ここでは Fernández-Villaverde et al. (2016) の定義を用いている。Judd (1998) では**ガラーキン法 (Galerkin method)** などの様々な残差の評価方法が紹介されている。

6　実際に手を動かそう

今回用いた例は全て GitHub からダウンロード可能である。「パラメータを変更したら何が変わるのか」を自分で試してみたりすることによって理解が進むことは多い。是非、自分でプログラムコードを動かしてみて、掲載されている図を再現してみてほしい。

コードの置き場所と使い方：

前回も述べたとおり、紙幅の都合上、プログラミング言語のインストール方法や基本的な使い方については説明できない。数値計算は特定の一つの言語と結びついているわけではないので、様々な選択肢があり、その全ての使い方を紹介するのは不可能だからである。また、プログラミング言語には様々な特徴があり、流行り廃れなどもあるので、著者たちがお気に入りの言語を押し付けるのは必ずしも適切ではないと考える。幸い一つの言語を習得すれば他の言語の学習コストは大幅に低下する。そのため、大学などで使えるか否か、あるいは自分の PC に導入する場合には予算の都合なども考慮しながら、お気に入りの言語を使って勉強していくのが良いであろう。

同様に、本連載で使用したコードの詳細を説明するのは紙面の都合で困難であるが、実際に用いたコードは原則として全て GitHub のサイト上に公開していく予定である。

- `https://github.com/TomoakiYamada/KeizaiSeminar_Chap2`

今回紹介した結果は全て MATLAB で書いている。例えば、上記の HP からコード一式をダウンロードして、`1_Discretization` というフォルダに入っている `main_discretization.m` というコードを実行すれば、3.2 節の結果を再現出来る。他節も同様にフォルダ内の `main_XX.m` と書かれたコードを走らせれば、結果を再現することが出来る。

コードに書かれた命令が数値計算のどの部分に対応しているかは、コード内に日本語でコメントが付けてある。コードを読むのはプログラミング言語を学習する上で非常に効果的な方法なので、是非、自力で解読してほしい。なお、著者たちが書いたコードは学習効果なども考慮して、必ずしも速度などの面で最適な書き方になっているわけではない。プログラミング言語と数値計算手法を勉強して、是非、より良いコードの書き方はないかを模索してほしい。

参考文献

Barro, Robert J. (1974) “Are Government Bonds Net Wealth?” *Journal of Political Economy*, Vol. 82, No. 6, pp. 1095-1117.

Browning, Martin, Lars P. Hansen, and James J. Heckman (1999) “Micro Data and General Equilibrium Models,” in Taylor, John B. and Michael Woodford eds. *Handbook of Macroeconomics, Edition 1*, Vol. 1: Elsevier, Chap. 8, pp. 543-633.

Cooley, Thomas F. and Edward C. Prescott (1995) “Economic Growth and Business Cycles,” in *Frontiers of Business Cycle Research*: Princeton University Press, Chap. 1, pp. 1-38.

Fernández-Villaverde, Jesús, Juan F. Rubio-Ramírez, and Frank Schorfheide (2016) “Solution and Estimation Methods for DSGE Models,” NBER Working Paper No. 21862.

Hansen, Lars P. and James J. Heckman (1996) “The Empirical Foundations of Calibration,” *Journal of Economic Perspectives*, Vol. 10, No. 1, pp. 87-104.

Judd, Kenneth L. (1998) *Numerical Methods in Economics*: The MIT Press.

Kydland, Finn E. and Edward C. Prescott (1996) “The Computational Experiment: An Econometric Tool,” *Journal of Economic Perspectives*, Vol. 10, No. 1, pp. 69-85.

Maliar, Lilia, Serguei Maliar, and Fernando Valli (2010) “Solving the Incomplete Markets Model with Aggregate Uncertainty using the Krusell-Smith Algorithm,” *Journal of Economic Dynamics and Control*, Vol. 34, pp. 42-49.

Song, Zheng, Kjetil Storesletten, and Fabrizio Zilibotti (2012) “Rotten Parents and Disciplined Children: A Politico-Economic Theory of Public Expenditure and Debt,” *Econometrica*, Vol. 80, No. 6, pp. 2785-2803.

ウィリアムソン，S. D. (2009) 『ウィリアムソン マクロ経済学 I 入門編』，東洋経済新報社.

バロー，R. (2010) 『バロー マクロ経済学』，センゲージラーニング.