2013年度夏学期　上級マクロ経済学講義 ノート:　動的計画法

阿部修人

平成25 年5 月10 日

1　動的計画法 (Dynamic Programming)

本章では動的計画法 (Dynamic Programming) の簡単な解説を行う。厳密な議論は避け、Bellman 方程式の意味および具体的な解法にについて解説する。より一般的、かつ厳密な議論に関しては Stokey and Lucas with Prescott(1987) を参照すること。

1.1　基本モデル

以下の単純な離散時間における経済成長モデルを考える。

$$\max_{\{c_t,k_{t+1}\}_o^T} \sum_{t=0}^{T} \beta^t U\left(c_t\right) \tag{1}$$

$$s.t. \quad c_t + k_{t+1} = Af\left(k_t\right), \tag{2}$$

$$k_0 \ : given, \vee t, \ k_t \geq 0, \ 0 < \beta < 1. \tag{3}$$

各変数の定義はこれまでの講義ノートと同様である。

この問題の解はどのように記述できるかを考えてみよう。上記の問題は条件付最適化問題であるが、(2) を目的関数に代入することにより、

$$\max_{\{k_{t+1}\}_o^T} \sum_{t=0}^{T} \beta^t U\left(Af\left(k_t\right) - k_{t+1}\right), \tag{4}$$

$$s.t. \quad k_0 : given, \ \vee t, k_t \geq 0. \tag{5}$$

の形に変形することが出来る。上記の最適化問題では資本ストックの系列に関して最適解を探せば良いことになる[1]。効用関数と生産関数が微分可能であり、十分滑らかであり、かつ内点解であると仮定すると、各期の資本ストックに関して偏微分し、一階条件式を得ることが出来る。

$$\beta^t U'\left(Af\left(k_t\right)-k_{t+1}\right)(-1)+\beta^{t+1}U'\left(Af\left(k_{t+1}\right)-k_{t+2}\right)Af'\left(k_{t+1}\right)=0 \quad (6)$$

これが t=1,2,3,...,T-1 に関して成立している。t=T、すなわち terminal point において、$k_{T+1}=0$ でなければならない。[2]

上の一階条件式は T 個の非線形の連立方程式である。未知数は k_1 から k_T までの T 個の資本ストックである。したがって、原理的には非線形連立方程式体系を解く問題とみなすことが出来る。しかしながら、T が大きな値をとるときには、この非線形連立方程式体系を解くことは非常に困難となる。

(1) を解析的に解く、すなわち、closed form の形で、各期の最適な資本ストックを得ることは一般的に不可能である。しかしながら、非常に特殊なケースに限り、closed fomr の形で解を得ることができる。

関数形として、以下の仮定をおく。

$$U\left(c_t\right)=\ln c_t \quad (7)$$

$$Af\left(k_t\right)=Ak_t^{\alpha}. \qquad 0<\alpha<1 \quad (8)$$

すると、一階条件は以下のようになる。

$$\beta^t\frac{1}{Ak_t^{\alpha}-k_{t+1}}(-1)+\beta^{t+1}\frac{\alpha Ak_{t+1}^{\alpha-1}}{Ak_{t+1}^{\alpha}-k_{t+2}}=0, \quad (9)$$

$$k_{T+1}=0. \quad (10)$$

ここで、新たな変数を導入し

$$X_t=\frac{k_{t+1}}{Ak_t^{\alpha}} \quad (11)$$

[1] より正確には、資本ストックが負にならないような制約が存在するが、生産関数および効用関数の両方に稲田条件を課せば、資本ストックのゼロ制約はバインドされるとはない。

[2] この理由は各自考えてみよ。最後の期に資本ストックを残してなにか良いことがあるだろうか?

とする。(9) の両辺に k_{t+1} を乗じ、整理すると一階条件は以下のようになる。

$$X_{t+1} = 1 + \alpha\beta - \frac{\alpha\beta}{X_t} \tag{12}$$

$k_{T+1} = 0$ であるから、$X_T = 0$ となる。したがって、

$$X_{T-1} = \frac{\alpha\beta}{1 + \alpha\beta - X_T} \tag{13}$$

さらに、

$$X_{T-1} = \frac{\alpha\beta}{1 + \alpha\beta} = \alpha\beta\frac{1 - \alpha\beta}{1 - (\alpha\beta)^2}. \tag{14}$$

$$X_{T-2} = \frac{\alpha\beta}{1 + \alpha\beta - \frac{\alpha\beta}{1+\alpha\beta}} = \alpha\beta\frac{(1 + \alpha\beta)}{1 + \alpha\beta + (\alpha\beta)^2} = \alpha\beta\frac{(1 + \alpha\beta)(1 - \alpha\beta)}{(1 - \alpha\beta)\left(1 + \alpha\beta + (\alpha\beta)^2\right)}. \tag{15}$$

したがって

$$X_{T-2} = \alpha\beta\frac{1 - (\alpha\beta)^2}{1 - (\alpha\beta)^3}. \tag{16}$$

繰り返すと、以下の公式を得ることが出来る。

$$X_t = \alpha\beta\frac{1 - (\alpha\beta)^{T-t}}{1 - (\alpha\beta)^{T-t+1}} \tag{17}$$

無論、この手法は terminal condition がある、すなわち最適化問題の視野が有限である (=T) ことに依存している。T が無限大であるとき、すなわち無限視野の場合は、この、後ろ向きに解く手法はそのままでは使うことが出来ない。

1.2 Bellman Equation and Policy Function

ここでは、無限視野の場合、すなわち $T = \infty$　のケースを考える。

$$\max_{\{k_{t+1}\}_o^\infty} \sum_{t=0}^{\infty} \beta^t \ln\left(Ak_t^\alpha - k_{t+1}\right). \tag{18}$$

この場合、terminal point がなくなり、$X_T = 0$ の条件を使えなくなる。したがって、前セクションの解法を直接用いることも不可能になる。また、一階条件式は無限個存在することになり、連立方程式体系として解

くことも非常に困難になる。このようなときは、もし解があるとした場合、それがどのような性質を持ちそうか、考えてみる。

(17) において、T を大きくしていくと、右辺は $\alpha\beta$ に収束していく[3]。したがって、t が十分に大きいとき X_t は $\alpha\beta$ にほぼ等しい、すなわち

$$k_{t+1} = \alpha\beta A k_t^{\alpha} \tag{19}$$

となることが予想される。(19) が正しいとすると、最適な資本ストック水準は前期の資本ストックのみに依存することになる。すなわち、各期の最適な資本ストックは

$$k_{t+1} = h\left(k_t\right), \quad t = 0, 1, 2, ... \tag{20}$$

の形で表されるのではないかと予想される。この関数 $h\left(k_t\right)$ は動的計画法では Policy Function と呼ばれている。このような Policy Function があり、各期の最適な資本ストックが前期の資本ストックのみに依存しているということがわかっていれば、無限の系列 $\{k_{t+1}\}_{t=0}^{\infty}$ をいきなり求める必要はなく、Policy Function を求めることができれば初期の資本ストック k_0 から容易に各期の最適な資本ストックを得ることができる。

さて、ここで (18) の解がわかっていると仮定しよう。そのときの最大化された生涯効用を初期の資本ストックのみに依存する関数とみなすと

$$V\left(k_0\right) = \max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \ln\left(A k_t^{\alpha} - k_{t+1}\right) \tag{21}$$

$V\left(k_0\right)$ は、k_0 の下で実現可能な最大の生涯効用水準を意味する。次に、時間を一期進めると、

$$V\left(k_1\right) = \max_{\{k_{t+1}\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} \ln\left(A k_t^{\alpha} - k_{t+1}\right) \tag{22}$$

で定義される $V\left(k_1\right)$ は、k_1 の下で実現可能な生涯効用になる。すると、t=0 時点における問題は

$$\max_{k_1} \left[\ln\left(A k_0^{\alpha} - k_1\right) + \beta V\left(k_1\right)\right] \tag{23}$$

となることが予想される。もしも $V\left(k_1\right)$ を知っていれば、上の問題を解くことが可能であり、最適な $k_1 = h\left(k_0\right)$ を得ることが出来る。なお、$V\left(k_0\right)$ を k_0 の下で実現可能な最大の生涯効用水準と定義していたので、

$$V\left(k_0\right) = \max_{k_1} \left[\ln\left(A k_0^{\alpha} - k_1\right) + \beta V\left(k_1\right)\right] \tag{24}$$

[3] $0 < \alpha\beta < 1$ による。

を得ることが出来る。期間をt期にすると

$$V(k_t) = \max_{k_{t+1}} [\ln(Ak_t^{\alpha} - k_{t+1}) + \beta V(k_{t+1})] \tag{25}$$

期間によらず、最適化問題が同一の構造をとっていることに注目すると、時間のindexを削除することが可能であり

$$V(k) = \max_{k'} [\ln(Ak^{\alpha} - k') + \beta V(k')] \tag{26}$$

を得ることが出来る。(26)はBellman方程式と呼ばれる。また、$V(k)$はValue Functionと呼ばれる。Value Functionがわかれば、その解としてPolicy Functionを得ることが出来、Policy Functionがわかれば、各期の最適な資本ストックの水準を得ることが出来るのである。無論、全ての動的最適化問題がこのような形で定式化できるとは限らないが、各期で繰り返し同一の問題に直面することが多い経済学の問題では応用範囲は広い。

1.3 一般ケース

Ljungqivist and Sargent (2004)に従い、各期のreturn functionをr、transition functionをgで表し、動的最適化問題を以下のように定義する。

$$\max_{\{u_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, u_t) \tag{27}$$

$$x_{t+1} = g(x_t, u_t) \tag{28}$$

$$x_0 : given. \tag{29}$$

上記の問題を解いて得た最適化された生涯効用をV^*であらわすと

$$V^* = \max_{\{u_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, u_t) \quad s.t. x_{t+1} = g(x_t, u_t) \quad \text{and } x_0 \tag{30}$$

そこで、Bellman方程式は以下のように定義される。

$$V(x_t) = \max_{u} (r(x, u) + \beta V(x')) \quad s.t. x' = g(x, u). \tag{31}$$

ここで問題になるのは、(1)Bellman方程式を満たす関数Vは存在するか否か、(2)存在するとき、それは一意か、それとも複数存在するか、(3)

その解はV^*と一致するか?である。もしもBellman方程式の解が一つしか存在せず、しかもV^*と一致すれば、無限視野の最適化問題は一期間の最適化問題を満たす関数Vを探す問題に変換することが可能になる。

残念ながら、常にBellman方程式の解がV^*と一致するとは限らない。また、Bellman方程式をみたす関数が複数存在する可能性もある。以下に、Bellman方程式がユニークな解をもち、それがV^*と一致するための十分条件を示す。詳しくはStokey and Lucas with Prescott(1987),

Chapter 4, Section 2を参照せよ。

(1)　rは強く凹かつ有界である。

(2)　集合$\left\{(x_{t+1}, x_t) : x_{t+1} \in g(x_t, u_t), u_t \in R^k\right\}$は凸かつコンパクトである。

(1)は、各期の効用関数が凹関数かつ、上限値があることを意味し、(2)は各期の生産可能集合が凸かつコンパクトであることを意味する。以上の条件の下でBellman方程式をみたす関数Vは以下の性質を持つ。

1. $V(x)$は単調増加関数である。
2. $V(x)$は強く凹であり、Policy functionはsingle-valued functionになる。
3. $V(x)$はV^*と一致する。
4. Bellman方程式の解Vは以下のiterationで得ることが出来る。

$$V_{j+1} = \max_u \left(r(x,u) + \beta V_j(x')\right) \quad s.t. x' = g(x,u). \tag{32}$$

5. 上記のiterationの収束先として得られたValue Function $V(x)$は以下の性質を満たす。

$$V'(x) = \frac{\partial r}{\partial x}(x, h(x)) + \beta \frac{\partial g}{\partial x}(x, h(x)) V'(g(x, h(x))). \tag{33}$$

5の性質はBenveniste and Scheinkmanの定理と呼ばれるものである。これは包絡線の定理とも呼ばれる。なぜなら、状態変数の一単位の増加は、それが瞬間効用関数rと生産可能性集合gを拡大させる直接効果のみがValue Functionに影響を及ぼし、Policy Functionの変化を通じた間接効果はValue Functionには影響を与えないことを示しているためである。

Benveniste and Scheinkmanの定理はクーンタッカーを用いた解との対比をするときに有効である。(26)の問題を考えてみよう。ここにBenveniste and Scheinkmanの定理を当てはめると

$$V'(k) = \frac{\alpha A k^{\alpha-1}}{A k^\alpha - k'}. \tag{34}$$

最適化の一階条件は

$$0 = \frac{-1}{Ak^{\alpha} - k'} + \beta V'(k') \tag{35}$$

$V'(k)$ の中が k と k' であることに注意してこの導関数を消去すると

$$-\frac{1}{Ak^{\alpha} - k'} + \beta \frac{\alpha A k'^{\alpha-1}}{Ak'^{\alpha} - k''} = 0. \tag{36}$$

ところで、DP ではなく、通常の動学問題として解くとオイラー方程式を下記のように得ることができる。

$$L = \sum_{t=0}^{\infty} \beta^t \ln\left(Ak_t^{\alpha} - k_{t+1}\right),$$
$$\beta \frac{A\alpha k_t^{\alpha-1}}{Ak_t^{\alpha} - k_{t+1}} - \frac{1}{Ak_{t-1}^{\alpha} - k_t} = 0$$

したがって、Benveniste and Scheinkman の定理から、通常のオイラー方程式を導くことができる。

条件 (2) は内生成長理論など、収穫逓増を含む生産関数を考えなければ通常は満たされる仮定である。しかし、(1) の扱いには注意が必要である。例で用いた対数効用関数には上限は存在せず、したがってこの条件 (1) を満たさない。実際、多くの場合、効用関数や利潤関数が有界であると仮定することはごく稀であり、標準的な経済モデルでは、この定理を用いることが出来ないのである。例えば、有名な Hall の消費のモデルでは効用関数が 2 次式と仮定しているが、この下では Bellman 方程式を満たす関数は複数存在することが知られている。Stokey and Lucas with Prescott(1987) の 4.4 節で有界でないケースを論じているが、マクロモデルで頻繁に用いられる関数形において、Value Function の一意性や最適性を保証することは通常非常に困難である。実際には、制約条件の凸、コンパクト性と効用関数が凹であることを確認し厳密な議論を避けて Value Function を用いることが多い。この厳密性の欠如が深刻な問題をつくるかどうかは今後の課題であるが、近年の論文によると、有界のケースを非有界のケースの多くの場合に拡張できるようである。

ここで興味深い性質は 4. である。最適化問題に解があることを示す定理は多いが、具体的にその解を導くアルゴリズムまで示すものは少ない。この iteration は Value Function Iteration と呼ばれるものであり、Dynamic Programming を解く際に中心的な役割を担うことになる。

1.4 Value Function Iteration

(32) で与えられたアルゴリズムを具体的にどう利用するか考える。まず、j=0 のとき、すなわち最初の時点での問題を考える。すると

$$V_1 = \max_u \left(r\left(x,u\right) + \beta V_0\left(x'\right) \right) \quad s.t. x' = g\left(x,u\right) \tag{37}$$

となる。右辺の最大化問題を解くには関数 V_0 を知る必要があるが、このアルゴリズムではとくに V_0 に関しては何も記述がない。上記の問題がもしも縮小写像になっていれば初期の V_0 は任意の関数でよいことが知られており、どんな関数、たとえば定数でも構わない。上記の問題を解いて得た関数 V_1 を右辺に持っていき、j=1 として

$$V_2 = \max_u \left(r\left(x,u\right) + \beta V_1\left(x'\right) \right) \quad s.t. x' = g\left(x,u\right) \tag{38}$$

と定義する。こうして V_j と V_{j+1} あるいはそれを生成する Policy Function $h_j\left(x\right)$ と $h_{j+1}\left(x\right)$ が十分に近くなったとき、すなわち iteration が違いを生み出さなくなるまでこのプロセスを続ける。そして得られた関数が求める Value Function および Policy Function、またはそれらの近似とみなすのである。

一部門の最適成長モデルを用いて、Value Function Iteration を実際に応用してみる。

まず $V_0 = 0$ を仮定する。すると、j=0 の問題は

$$V_1 = \max_{k'} \ln\left(Ak^\alpha - k'\right) + \beta \times 0. \tag{39}$$

この問題の解は $k' = 0$ のとき、

$$V_1 = \ln Ak^\alpha = \ln A + \alpha \ln k \tag{40}$$

となる。つぎに、j=1 として、この V_1 を用いて

$$V_2 = \max_{k'} \ln\left(Ak^\alpha - k'\right) + \beta\left(\ln A + \alpha \ln k'\right) \tag{41}$$

とする。この最適化問題の一階条件は

$$\frac{-1}{Ak^\alpha - k'} + \beta\alpha\frac{1}{k'} = 0 \tag{42}$$

である。整理すると

$$k' = \frac{\alpha\beta}{1+\alpha\beta} Ak^\alpha \tag{43}$$

であり、これを代入すると

$$V_2 = \ln\left(\left(1 - \frac{\alpha\beta}{1+\alpha\beta}\right)Ak^{\alpha}\right) + \beta\left(\ln A + \alpha \ln \frac{\alpha\beta}{1+\alpha\beta}Ak^{\alpha}\right) \tag{44}$$

したがって

$$V_2 = \ln \frac{A}{1+\alpha\beta} + \beta \ln A + \alpha\beta \ln \frac{\alpha\beta}{1+\alpha\beta} + \left(\alpha + \alpha^2\beta\right)\ln k \tag{45}$$

さらにこれを用いて j=2 とすると

$$V_3 = \max_{k'} \ln\left(Ak^{\alpha} - k'\right) + \beta\left(Const1 + \left(\alpha + \alpha^2\beta\right)\ln k'\right) \tag{46}$$

この一階条件は

$$\frac{-1}{Ak^{\alpha} - k'} + \beta\left(\alpha + \alpha^2\beta\right)\frac{1}{k'} = 0 \tag{47}$$

であり、

$$k' = \frac{\alpha\beta\left(1+\alpha\beta\right)}{1+\alpha\beta\left(1+\alpha\beta\right)}Ak^{\alpha} \tag{48}$$

が解となる。V_3 は

$$V_3 = Const2 + \alpha\left(1 + \alpha\beta + \left(\alpha\beta\right)^2\right)\ln k' \tag{49}$$

となる。これを繰り返していくと、

$$V\left(k\right) = \frac{1}{1-\beta}\left(\ln\left(A\left(1-\alpha\beta\right) + \frac{\alpha\beta}{1-\alpha\beta}\ln\left(A\alpha\beta\right)\right)\right) + \frac{\alpha}{1-\alpha\beta}\ln k \tag{50}$$

を得る。これが正確な Value Function であり、付随する Policy Function は

$$k' = \alpha\beta k^{\alpha} \tag{51}$$

である。

この例からわかるように、Value Function は手法としては単純であるが、収束が遅い。上記の例では lnk の係数は等比数列の和として表れ、その無限和が所望の Value Function となる。もう少し速く収束させる、または容易に Value Function を求める手法がいくつか開発されている。

1.5 Guess and Verify

もっとも単純なDPの解法はGuess and Verifyである。これは、Value FunctionをGuessし、その関数がBellman Equationを満たすことを確認する、すなわちVerifyするというものである。[4]これはValue Functionの形状についてある程度の知識があることが前提となる。残念ながら、この手法の応用範囲は狭く、Closed Formで解が存在するものに限られる。

再び、(18)を用いてGuess and Verifyを実践してみよう。効用関数が対数であることから、Value Functionも対数であるとGuessしてみよう。すなわち、

$$V(k) = E + F \ln k \tag{52}$$

ただし、EとFは定数である。ここで、このValue FunctionがBellman Equationを満たす、すなわち

$$E + F \ln k = \max_{k'} \ln (Ak^{\alpha} - k') + \beta (E + F \ln k') \tag{53}$$

であることをVerifyすることができればよい。最適化の一階条件は

$$0 = \frac{-1}{Ak^{\alpha} - k'} + \frac{\beta F}{k'} \tag{54}$$

である。したがって

$$k' = \frac{\beta F A k^{\alpha}}{1 + \beta F} \tag{55}$$

となる。これをBellman Equationに代入して整理すると

$$E + F \ln k = \ln \frac{A}{1 + \beta F} + \alpha \ln k + \beta E + \beta F \ln \frac{\beta F A}{1 + \beta F} + \alpha \beta F \ln k \tag{56}$$

これが恒等式であるとすると

$$F = \alpha + \alpha \beta F, \tag{57}$$

$$E = \ln \frac{A}{1 + \beta F} + \beta E + \beta F \ln \frac{\beta F A}{1 + \beta F}. \tag{58}$$

これをE,Fに関する連立方程式と考えると

$$F = \frac{\alpha}{1 - \alpha \beta} \tag{59}$$

[4]Long and Plosser(1983)はGuess and Verifyを用いてDynamic Programmingを解いている。

$$E = \frac{1}{1-\beta}\left(\ln\frac{A}{1+\beta F} + \beta F \ln\frac{\beta F A}{1+\beta F}\right) \tag{60}$$

を得る。E と F に関して解ききったので、この関数は Bellman Equation を満たすことがわかり、この Guess は正しかったことが証明されたことになる。

この手法が実際に応用可能なモデルは数えるほどしかなく、あまり実用的ではない。また、どのような関数を Guess するかに関しても決まった手法はなく、一種の技法として考えるしかない。

1.6 Policy Function Iteration

これは Value Function の代わりに Policy Function を iterate する手法であり、Value Function よりも速く収束することが多いと言われている。別名 Howard's Policy Improvement Algorithm とも言われる。

1. まず、実行可能な Policy Function の候補

$$u = h_0(x_0) \tag{61}$$

を適当にとる。

2. つぎに、この Policy Function により得られる Value Function の候補を計算する。すなわち

$$V_0(x_0) = \max_{\{u_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, h_0(x_t)) \quad s.t. x_{t+1} = g(x_t, h_0(x_t)) \quad \text{and } x_0. \tag{62}$$

3. 上で得られた Value Function の下で、新たに Policy Function を計算する。すなわち

$$\max_u \left(r(x,u) + \beta V_0(x')\right) \quad s.t. x' = g(x,u) \tag{63}$$

を解く。

4. また 1. から繰り返し、Policy Function があまり変化しなくなるまで続ける。

再び、(18) を用いて Policy Function Iteration を実践してみる。

1. まず、実行可能な Policy Function を適当に推測する。例えば、貯蓄性向が 1/2 であると仮定し

$$k_{t+1} = \frac{1}{2} A k_t^{\alpha} \tag{64}$$

とする。

2. これを効用関数に代入し、生涯効用、すなわち Value Function の候補を計算する。

$$V_0(k_0) = \sum_{t=0}^{\infty} \beta^t \ln\left(A k_t^{\alpha} - \frac{1}{2} A k_t^{\alpha}\right) \tag{65}$$

$$= \sum_{t=0}^{\infty} \beta^t \ln\left(\frac{1}{2} A k_t^{\alpha}\right) \tag{66}$$

$$= \sum_{t=0}^{\infty} \beta^t \left(\ln\left(\frac{1}{2} A\right) + \alpha \ln k_t\right) \tag{67}$$

ここで、

$$k_t = \frac{1}{2} A k_{t-1}^{\alpha} \tag{68}$$

$$= \left(\frac{1}{2}\right)^{1+\alpha} A^{1+\alpha} k_{t-2}^{\alpha^2} \tag{69}$$

であることを利用して

$$k_t = \ln D + \alpha^t k_0 \tag{70}$$

ただし、D は定数である。したがって、

$$V_0(k_0) = \sum_{t=0}^{\infty} \beta^t \left(\ln\left(\frac{1}{2} A\right) + \alpha \ln D + \alpha^{t+1} \ln k_0\right) \tag{71}$$

$$V_0(k_0) = const + \frac{\alpha}{1 - \alpha\beta} \ln k_0. \tag{72}$$

3. 上で求められたValue Functionを用いてBellman Equationに戻ると

$$V_0(k) = \max_{k'} \ln(Ak^\alpha - k') + \beta\left(const + \frac{\alpha}{1-\alpha\beta}\ln k'\right) \tag{73}$$

この一階条件を用いると

$$\frac{-1}{Ak^\alpha - k'} + \frac{\alpha\beta}{1-\alpha\beta}\frac{1}{k'} = 0 \tag{74}$$

したがって

$$k' = \alpha\beta Ak^\alpha \tag{75}$$

となり、一回のiterationで正しい解が得られたことになる。

Polify Function Iterationの方が収束が早いことを証明した研究は、筆者の知る限りない。しかしながら、筆者の経験からくる直感的理由は下記のようなものである。最適化問題を解くとき、Value Functionが求まれば、Policyを求めることができる。同様に、PolicyからValue Functionを定義することも可能であり、数学的にはどちらを求めても同じ筈である。しかしながら、求める精度が同じであれば、例えば0.0001の精度でValue Functionを求める場合と、0.00001の精度でPolicyを求める場合にかかる時間は後者のほうが短いケースが多い。Policyをごくわずか変更しても、Valueの値は大きく変化してしまうことがある。Valueの値はPolicyの累積であるためである。逆をいうと、Valueが求まれば、その近傍でのPolicyはほぼ同じものとなる。ある精度のPolicyを求めることは、より高い精度のValueを求めることと同じになるのである。

2 数値計算: Discretization (本節は中間試験範囲には含まれない)

これまで紹介したValue Function Iteration、Guess and Verify、およびPolicy Function Iterationは、いずれも正しい解を手計算で得ることができた。しかしながら、これが可能だったのは各ステップでValue FunctionやPolicy Functionの候補をState Variableの関数として、Closed Formで表現することができたためである。残念ながら一般にClosed Formの

解は存在せず、これらの手法は使えない。Closed Form では記述できない関数を、形で、扱いやすい関数で近似せねばならない。

Closed Form が存在しない関数を我々の既知の関数で近似する場合、大きく分けて 2 つの手法が考えられる。一つは多項式、あるいはスプライン関数で近似する手法であり、もう一つは関数ではなく、点の集合として扱う手法である。前者の中には良く知られている線形近似も含まれる。前者の場合、既知である多項式の集合の中で、Closesd Form のない価値関数 (あるいは Policy Function) に最も近いものを探すことになる。換言すれば、多項式のつくる空間に直行写像を作るということであり、それゆえ Projection Methods と言われる。

後者は一般に離散近似法と呼ばれるものであり、多項式という制約を置かずに解くことが可能であるため非常に強力である。特に、数値計算に用いられる多項式の多くが連続微分可能な Class に限定される、すなわち価値観数も連続微分可能なものと仮定せざるを得ないのに対し、離散近似の場合連続微分可能とは限らない価値観数や Policy Function も扱うことが出来る。これは、非可逆的投資や借り入れ制約など、今日の経済で興味深い問題、広範囲な分析対象をもつという大きな利点がある。一方、数値計算に必要な Computation の負荷もまた多項式近似に比べて大きく、メモリーサイズの制約から、多くの状態変数を持つシステムの解法には向かないという欠点もある。とくに、資本ストックも含む国際貿易や多部門モデルへの応用は現時点ではまず不可能であろう。

本節では、離散近似法を紹介する。

3　最適成長モデル

「離散近似」という言葉に含まれる “離散” は、状態変数を連続な数値ではなく、有限個の点の集合とみなすという意味である。標準的なの最適成長モデルでは、資本ストックと技術水準が状態変数であり、$(0,\infty)$ の実数値をとると仮定していたが、離散近似の場合、例えば {0.1, 0.2, 0.3, ...,0.9, 1} の値しかとらないと仮定するのである。無論、この区間を広く取れば取るほど、また点を細かく取れば取るほど正確な解に近くなっていく。一方、点を多く取れば取るほど計算にかかる時間とメモリーの量が膨大になっていく。これは Curse of Dimension と呼ばれる現象であり、後に説明する。

前節までと同様に、単純な最適経済成長モデルを考えよう。ただし、今

回は対数効用ではなく、CRRA を仮定し、Closed Form での解が存在しないモデルを考える。また、減価償却のない世界を考える。無論、これらの制約を緩めることは極めて簡単である。

$$\max_{\{c_t,k_{t+1}\}_o^\infty} \sum_{t=0}^{\infty} \beta^t \frac{c_t^{\gamma+1}}{\gamma+1} \tag{76}$$

subject to

$$k_{t+1} = Ak_t^\alpha + k_t - c_t \tag{77}$$

定常状態での資本ストックが 1 になるようにパラメター A を調整すると、定常状態においては

$$Aak^{\alpha-1} + 1 = \frac{1}{\beta} \tag{78}$$

であるから

$$A = \frac{1-\beta}{\alpha\beta} \tag{79}$$

とする。Judd [1998] の例に従い、パラメターを $\gamma = -2, \alpha = 0.25, \beta = 0.96$ とする。

1. まず、資本ストックをどのような集合で近似するかを決定する。ここでは、定常状態の水準を 1 としたので、その区間を含む集合を考える。例えば、0.2 と 1.8 の間を 0.001 の間隔で埋めると[5]、資本ストックは 1600 個の点の集合となる。

2. 次に、来期の資本ストックと今期の資本ストックの組み合わせを考える。予算制約より

$$c_t = Ak_t^\alpha + k_t - k_{t+1} \tag{80}$$

であり、今期の資本ストックと来期の資本ストックの全ての組み合わせを考えることで、消費量を決定することが出来る。各期の効用関数を資本ストックで定義し

$$u = u\left(k_t, k_{t+1}\right) \tag{81}$$

[5]この例は非常に多くのメモリーを要求するが、それでも 2GHz の PentiumIV に 768MB のメモリーを積んだ PC では数分で計算を終えることが出来る。

と考えることと等しい。予算制約を使って、今期の資本ストックと来期の資本ストックの可能な組み合わせ、ここでは 1600×1600 の行列の各要素に消費水準を割り当てる。なお、グリッドの数を増やすと、より広い範囲で、またはより詳細な形状を知ることが可能であるが、計算の時間の増大もまた著しい。1600 というのは、現在の

3. Value Function Iteration を行う。Value Function の初期値として、例えば zero 関数を用いる。すなわち、

$$V_1 = \max_{k_{t+1}} \left(u\left(k_t, k_{t+1}\right) + \beta V_0\left(k_{t+1}\right) \right) \tag{82}$$

を考える際に、$V_0 = 0$ を初期値とする。ここで、$u\left(k_t, k_{t+1}\right)$ は 1600×1600 の行列であり、max をとるということは、今期の資本ストックの水準、ここでは 1600 個の水準ごとに、右辺の値を最大にするような来期の資本ストックを 1600 の中から一つ選ぶことに等しい。最初のステップでは Value Function の初期値が zero であるから単に効用関数を最大化する組み合わせが選ばれ、次のステップでは、得られた Value Function を右辺に移動して、また今期の各資本ストックごとに、右辺全体を最大化させる来期の資本ストックが選ばれる。

4. Value Function が収束するまで Iteration を繰り返す。ここでは、Value Function の存在などは大きな問題にならない。なぜなら資本ストックの水準に上限および下限を与えているため、各期の効用関数は有界であり、また凹関数になっているので Value Function は Unique に決まり、かならず収束する。

5. 得られた Value Function から Policy Function を計算する。ここでいう Policy Function は、Iteration の最後で選ばれた今期の資本ストックと来期の資本ストックの組み合わせに他ならない。

では、具体的に Matlab のプログラムを見てみよう。

4 Matlab Code

```
% パラメターの設定
%
```

```
alpha = 0.25; % production parameter
beta = 0.9; % subjective discount factor
delta = 1; % 1 -depreciation rate
gam = -2; % preference parameter
%
%
% 資本ストック水準の離散化
%
mink = 0.2; % minimum value of the capital grid
maxk = 1.8; % maximum value of the capital grid
ink = 0.001; % size of capital grid increments
nk = round((maxk-mink)/ink+1); % number of grid points
%
% 各期の効用関数の作成
%
util = -1000000*ones(nk,nk);
%
%   効用の期値を極めて低い値にすることで、初期状態が選択される
%   ことを防いでいる。例えば、今期の資本ストックが小さいとき、来
期の
%
%
% 今期の資本ストックに関するループ
%
for i=1:nk;
    k=(i-1)*ink + mink;
    %
    % 来期の資本ストックに関するループ
    %
    for j=1:nk;
        kprime = (j-1)*ink + mink;
        invest = kprime - delta*k;
        cons = ((1-beta)/(alpha*beta))*(k^(alpha)) - invest;
        if cons > 0;
            util(j,i)= (cons^(gam+1))/(gam+1);
```

```
                % 今期の資本が i で、来期資本が j の効用水準
            end;
        end;
    end;
    %
    % 各関数の初期化
    %
    v = zeros(nk,1); % Value Function の初期値
    % Value Function の定義域は今期の資本ストックの各グリッドである
    % ことに注意。%
    decis = zeros(nk,1);
    iter = 0;
    metric1 = 10;
    metric2 = 10;
    [rs,cs] = size(util);
    %
    % Bellman Equation を iterate する。
    %
    while metric1 ~= 0 | metric2 > 0.001 ;
       %    metric2 を 0.01 とすると 60 回で、0.001 とすると
        %    76 回の iteration で、metric2 ~= 0 とすると 381
        %    回で収束する。0.01 以下にしてもあまり結果に
        %    違いは出ない。
        %
        % すべての state に関してループを走らせる。
        %
        for i=1:cs;
            r(:,i) = util(:,i) + beta*v;
        end;
        [tv,tdecis] = max(r); % tv is 1*cs vector with max of each cols of
r
        % tdecis は 1*cs ベクトルであり、この要素は r の値を最大にする
ものである。
        % この関数は各状態において Value Function を所与として
```

```
        % Bellman Equation を最大にする来期の資本ストック水準を与
える。
        % tv はそのときの value の値である。
        %
        tdecis = tdecis';
        tv = tv';
        metric1 = max(any(tdecis-decis)); % test nonzeros (1 if yes, 0 OW)
      % Policy Function の違いのチェック
        metric2 = max(abs(v-tv));
      % Value Function の違いのチェック
        v = tv;
        decis = tdecis;
        vfor8 = v(1); % value of v for k=kmin=0.2
        ufor8 = (decis(1)-1)*ink + mink; % value of control for k=0.2
        iter = iter+1;
        disp([ iter metric1 metric2 vfor8 ufor8]);
      % iteration の様子の表示
  end;
  %
  %Policy Function と Value Function を計算する。
  %
  policy = (decis-1)*ink + mink; % policy function
  %
  p = -10000*ones(nk,1); % utility under the optimal policy
  for i=1:cs;
        k = (i-1)*ink + mink;
        invest = policy(i) - delta*k;
        cons = ((1-beta)/(alpha*beta))*(k^(alpha)) - invest;
        if cons > 0;
              p(i)= (cons^(gam+1))/(gam+1);
        end;
  end;
  betam = beta*ones(cs,1);
  value = p/(ones(cs,1)-betam);
  %
```

```
% 結果を出力する。
%
disp('PARAMETER VALUES');
disp('');
disp(' alpha beta gamma ');
disp([ alpha beta gam ]);
disp('');
%
%
% 結果をグラフに出力する。
%
kgrid = [(mink:ink:maxk)']; % capital grid
%
figure(1)
plot(kgrid',value);
title('DETERMINISTIC GROWTH MODEL: VALUE FUNCTION');
%
figure(2)
plot(kgrid',policy,'.',kgrid',kgrid','-');
title('DETERMINISTIC GROWTH MODEL: POLICY FUNCTION');
```

図1はこの結果得られるValue Functionを示している。Value Functionは単調増加の凹関数になっており、Lucas, Stokeyの結果と整合的である。また、図2で示されるPolicy Functionは極めて線形に近く、わずかに45度線よりも小さい傾きになっていることがわかる。また、定常状態が1であることも、Policy Functionが1で45度線と交わっていることから確認できる。

一つ興味深いのは、この離散近似の結果と線形近似の結果との比較であろう。線形近似の場合、Value FunctionもPolicy Functionも線形と仮定されている。離散近似の場合のValue Functionが凹になっているため、定常状態から乖離した点、とくに左側では線形近似と離散近似の乖離は大きくなるであろう。また、線形近似の場合はテイラーの定理より、定常状態付近では両者はかなり近くなることが予想される。図3は両者の乖離を示している。予想通り、資本ストックがゼロに近いところでは両者の乖離は大きくなっている。縦軸のスケールは0.001であり、Value Functionの値は-40近くであるから、両者の乖離はそれほど大きくはない。また、

興味深いのは、両者の乖離が波を打っていることである。これは離散近似の場合常に生じる現象であり、これを回避するには grid をさらに細かくとっていくしかないように思われる。

下記のコードを前記の離散近似 DP の後半に置くことで、線形近似と離散近似の解を比較することが出来る。

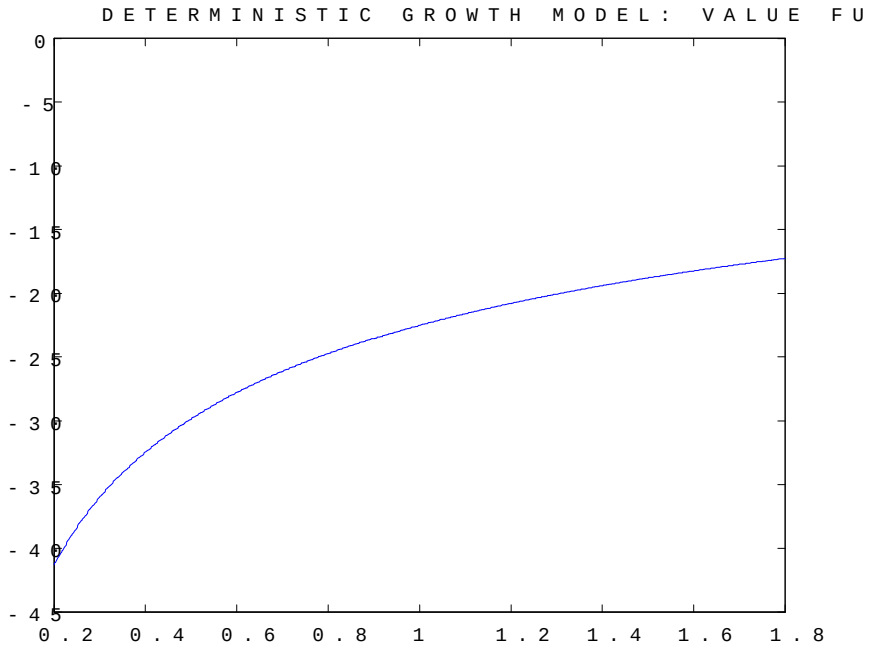
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cons = (1-beta)/(alpha*beta);
lam = cons^gam;
%
n=1; % The number of the predtermined variables
%
%
%
%
% Matrices For subroutine to solve dynamic optimization problem
%
%
% MCC matrix
%
mcc=zeros(1,1);
mcc(1,1) = gam * cons^(gam-1);
%
%
% MSC Matrix
%
mcs = zeros(1,2);
mcs(1,2) = 1;
%
%
% MCE Matrix - no stochastic elements
%   不確実性はないからゼロ行列
%
mce = zeros(1,1);
%
```

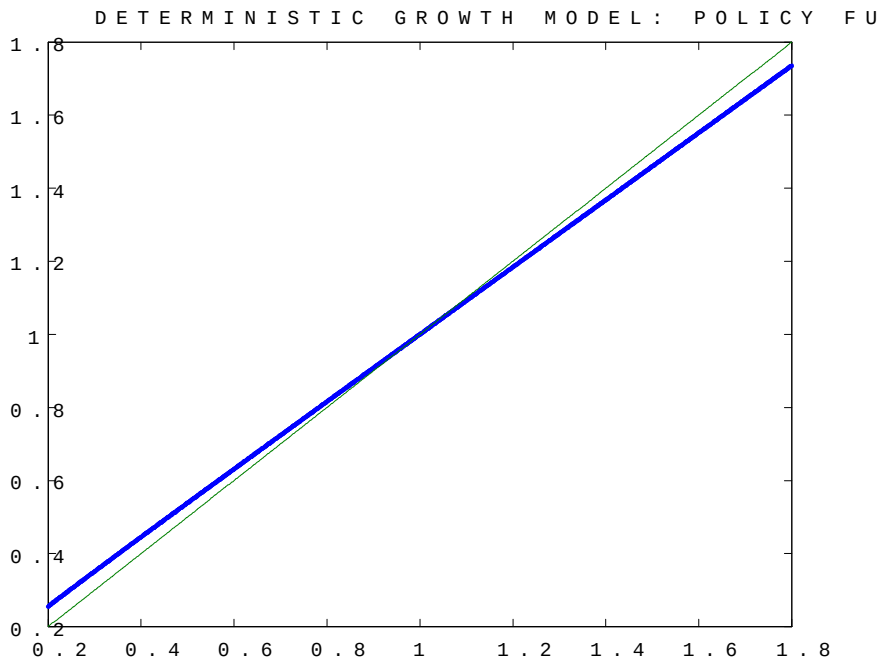
```
%
% MSS0 Matrix
%
mss0 = zeros(2,2);
mss0(1,1) = lam*(1-beta)*(alpha-1);
mss0(1,2) = 1;
mss0(2,1) = 1;
%
%
% MSS1 Matrix
%
mss1 = zeros(2,2);
mss1(1,2) = -1;
mss1(2,1) = -1/beta;
%
%
% MSC0 Matris
%
msc0 = zeros(2,1);
%
%
% MSC1 Matrix
%
msc1 = zeros(2,1);
msc1(2,1) = -1;
%
%
% MSE0 Matrix
%
mse0 = zeros(2,1);
%
% MSE1 Matrix
%
mse1 = zeros(2,1);
%
```

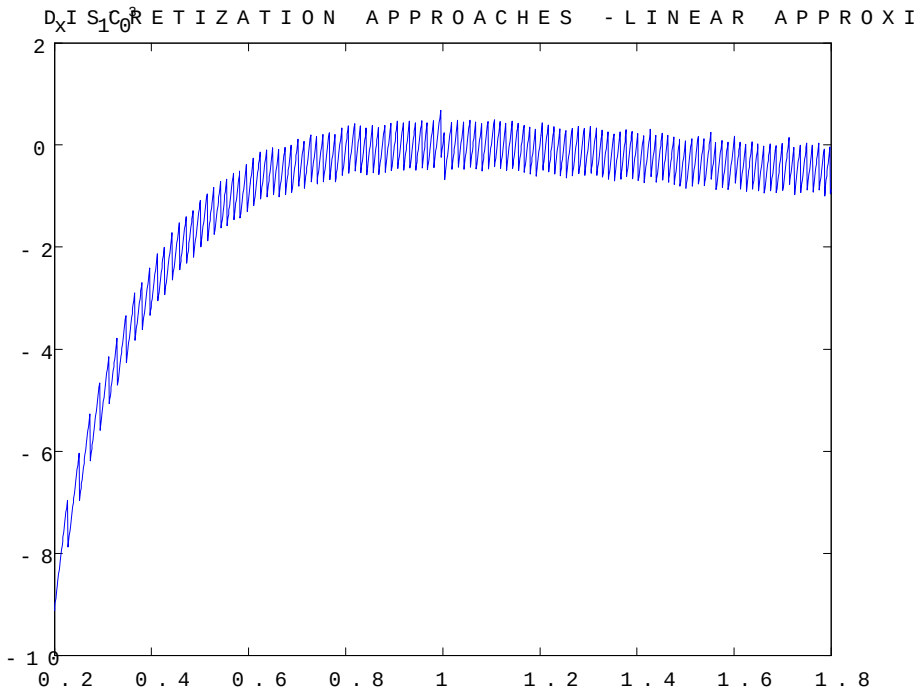
```
%
% PAI Matrix
%
pai = zeros(1,1);
%
%
%
[GXX,GXZ,GUX,GUZ,M,Psi,V] = burns6(n,mcc,mcs,mce,mss0,mss1,msc0,msc1,mse0,mse1,p
%
%
linpol = zeros(cs, 1);
for i = 1:cs
      linpol(i) = 1-GXX +GXX*(mink+ink*(i-1));
end
diff = policy-linpol;
%
figure(3)
plot(kgrid',diff);
title(' DISCRETIZATION APPROACHES -LINEAR APPROXIMA-
TION(400)');

% % % % % % % % % % % % % % % %5
```

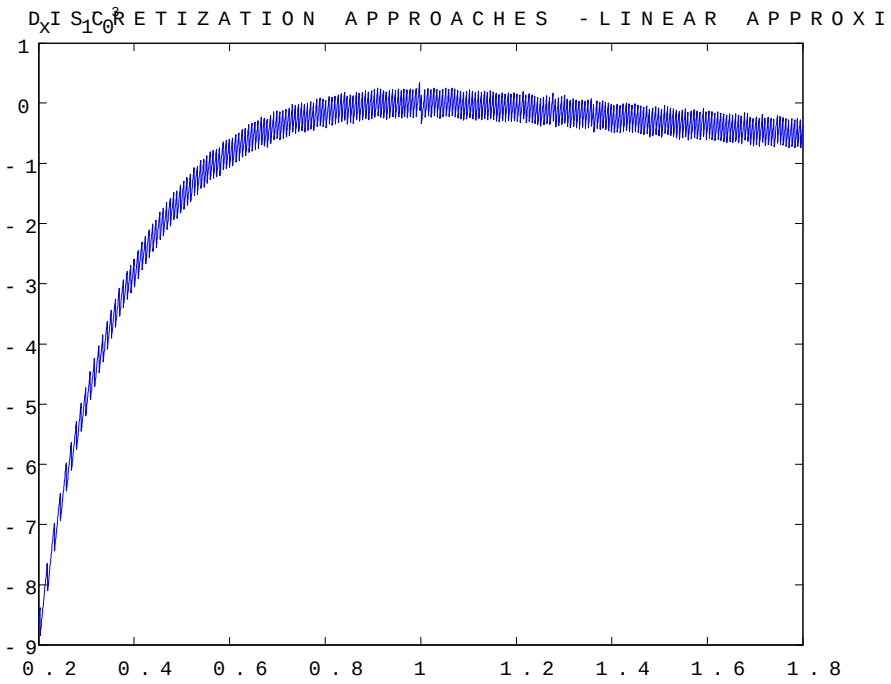
なお、上記の計算で示したように、ここでは Value Function Iteration の中で 1600 × 1600 の行列を二つ用いている。各行列の要素に数字が入るわけであり、必要メモリーは数十メガバイトとなる。さらに grid の数を増やすと、必要メモリーはその二乗で増えることになり、より高い精度を求めると、PC のメモリーがすぐに不足する。特に、State Variable の数が増えると、例えば現在は 1 つであるが、2 つになると、必要メモリーはさらにその二乗となり、実質的に解くことは不可能になる。これが Curse of Dimension と呼ばれる現象であり、コンピュテーションの限界が、スピードではなく、メモリーによるものとなり、たとえ大型の汎用機や Super Computer を用いたとしても解決できない深刻な問題となっている。

ちなみに、上記の例では、Maltab 全体が PC で使用するメモリーは 500MB ほどで、^変数 r には 20505608 バイトが割り振られ、実行時間は、Laptop PC で 20 秒ほどであるが、精度を倍にし、3200 のグリッドで近似


DETERMINISTIC GROWTH MODEL: VALUE FU
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8


DETERMINISTIC GROWTH MODEL: POLICY FU
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8


DISCRETIZATION APPROACHES - LINEAR APPROXI
1e-03
2
0
-2
-4
-6
-8
-10
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8



した場合、使用メモリーは 940MB で変数 r には 81971208 バイトが割り振られ (先ほどの 4 倍)、時間は 340 秒となる。