目次

7 2 期間最適消費モデル 2
7.1 概要 2
7.2 理論: ミクロ経済学の復習 3
7.2.1 効用関数 3
7.2.2 予算制約 5
7.2.3 効用最大化と均衡 5
7.2.4 ラグランジュ未定乗数法 7
7.3 理論: 2 期間の最適消費モデル 10
7.3.1 価値の割引 10
7.3.2 効用の割引 11
7.3.3 予算制約 12
7.3.4 解法 14
7.3.5 解の性質 15
7.4 プログラミング 17
7.4.1 シンボリック計算と数値計算 18
7.4.2 シンボリック計算 18
7.4.3 数値計算 24
7.4.4 シンボリック計算と数値計算のあわせ技 29

7 2期間最適消費モデル

7.1 概要

第**??**章のソロー・モデルの分析では，当期所得の一定割合 $(1-s)$ が消費され，残り s が貯蓄されると仮定した：

$$C_t = (1-s)Y_t, \qquad t = 1, 2, \ldots$$

この仮定には次のような批判があるだろう。消費は当期の所得だけでなく，将来所得にも影響されるはずだ。を将来の所得が高いと予想している個人は，現在借入れをしてでも消費を増やすだろうし，将来所得が小さくなると予想している個人は貯蓄を増やして消費を減らす行動を取るだろう。

ソロー・モデルを使うと，1 人あたり実質 GDP の長期的な成長率を決定する要因が技術進歩であることを示すことができる。この結果は基本的な成長会計とも整合的であるし，カルドアの事実のほとんどを説明する。使いやすいモデルではあるのだが，その良好な性能が「将来所得を考慮せずに当期の所得を決める」という強い仮定にあるのだとすれば，このモデルに対して,「たまたまうまくいっただけ」,「データに合うように都合よくモデルを作っているだけ」と考える人も現れるだろう。経済学者は最適化問題の解として説明できない仮定を受け入れないように訓練されているのだ。

この章から数章にわたって，消費者の意思決定の方法に関する仮定を緩めてもソロー・モデルの基本的な結論に影響しないことを説明する。導入部となるこの章では，もっとも単純な，代表的個人が 2 期間の消費計画を最適に決定する問題を紹介する。代表的個人とは，要するに，経済に人が 1 人しかいないということだ。そのような経済をイメージしにくければ，同じ選好を持つ多数の消費者がいる経済だと考えてもよい。この経済には獲得した財を貯蔵して増やす手段があって，1 期と 2 期の間で消費の配分を選ぶことができる。1 期にたくさん消費すると，2 期には消費を減らさなければならないので，消費者は**異時点間の資源配分**の問題に直面しているのだ。

この章で学ぶモデルは重要なモデルの基礎となる。計画期間を 2 から無限大に伸ばし，無限期間生きる代表的個人を考える。さらに，ソロー・モデルと同じ資本蓄積の方程式と合わせると，最適成長モデルあるいはラムゼー・モデルと呼ばれる成長モデルが得られる。計画期間を伸ばす代わりに，2 期間の最適化問題を解く経済主体が毎期生まれて，2 期生きたあとに死んでいくようにすると，世代重複モデルと呼ばれるモデルになる。

本章では，

7.2 理論: ミクロ経済学の復習

7.2.1 効用関数

異なる財を同時期に消費する計画を立てる静学的な問題から出発しよう。経済には財 1, 2 があって，消費量の配分を検討しているとしよう。消費量（より正確に言えば消費しようと計画する量）をそれぞれ x_1, x_2 のように下付き添字 1, 2 で表現する。消費者は「効用関数」（utility function）という選択の基準を持っているとする。効用関数は計画消費量 x_1, x_2 の関数として，

$$U(x_1, x_2)$$

と書く。U は計画消費量の組合せ (x_1, x_2) に点数を付けていく関数で，財 1, 2 ともに「消費量が増えるとうれしい」という普通の財であれば [1]，

$$x_1' \geq x_1 \quad \text{and} \quad x_2' \geq x_2 \quad \text{どちらか一方は厳密な不等式}$$

であるとき，

$$U(x_1', x_2') > U(x_1, x_2)$$

が成り立つ。これは「(x_1, x_2) 平面を右上に行けば行くほど効用が高まる」ことを表している。

その他もろもろの技術的な仮定を置くのだが，詳細は適当なミクロ経済学の教科書を参照してもらいたい。分析上で特に重要な仮定は，無差別曲線の形状に関するものである。ある点 (x_1', x_2') を通る無差別曲線は，

$$\left\{(x_1, x_2) \in \mathbb{R}^2_+ \mid U(x_1, x_2) = U(x_1', x_2')\right\}$$

なる集合のことである [2]。取りうる効用水準すべてについて無差別曲線が定義できるので，(x_1, x_2) 平面に無数の無差別曲線曲線を描くことができる。無差別曲線には次のことを仮定する。すべての無差別曲線が，

- 厚みのない「線」である，
- 他の無差別曲線と交わることがない，
- 原点に向かって厳密に凸である，
- なめらかな線である。

このような仮定の下で，無差別曲線は図 7.1 の原点に向かって凸な曲線のようになる。本当は無数に描けるのだが，3 本だけ描いていることに注意してほしい。右上に位置している無差別曲線の方が，左下に位置しているものよりも効用水準が高い。

[1] 財（goods）は良い（good）ものである。消費量を増やすと効用が下がる bads という概念もある。例えば，金融資産のリスクとか，大気汚染物質とか，労働とか。

[2] $\mathbb{R}^2_+$ は 2 次元空間（平面）の第 1 象限を表す。すなわち，$\mathbb{R}^2_+ = \left\{(x, y) \in \mathbb{R}^2 \mid x \geq 0,\ y \geq 0\right\}$.

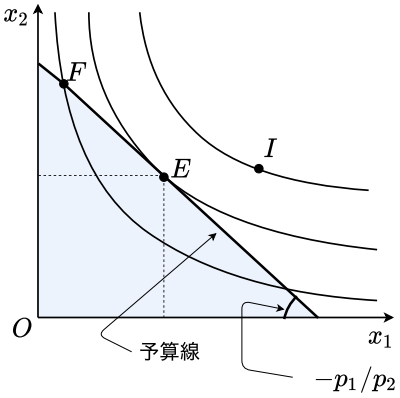


図 7.1: 無差別曲線と最適消費点

任意の点 (x_1', x_2'), $U(x_1', x_2') = \bar{U}$, の近くに無差別曲線

$$\{(x_1, x_2) \mid U(x_1, x_2) = \bar{U}\}$$

の接線を引くことができる。接線の傾き dx_2/dx_1 を求めるには，全微分を用いて次のような形式的な操作をするとよい [3]。

$$dU = \frac{\partial U}{\partial x_1} dx_1 + \frac{\partial U}{\partial x_2} dx_2$$

無差別曲線 「U = 一定」を保つ方向の微分を計算したいので，$dU = 0$ とすると，

$$\frac{dx_2}{dx_1} = -\frac{\left(\frac{\partial U}{\partial x_1}\right)}{\left(\frac{\partial U}{\partial x_2}\right)} = MRS$$

を得る。接線の傾きを**限界代替率**（marginal rate of substitution, MRS）と呼ぶ（図 7.2）。財 1 の消費を増やしたときに，財 2 の消費を減らさなければ効用が変化してしまう。効用を一定に保つために財 2 を減らさなければならない量が限界代替率である。効用水準を変えずに，財 2 から財 1 に代替しているのである。このことは全微分の公式に戻ってみると分かりやすい。

$$\frac{\partial U}{\partial x_1} \times 1 + \frac{\partial U}{\partial x_2} \times \underbrace{\left(-\frac{\partial U/\partial x_1}{\partial U/\partial x_2}\right)}_{=MRS} = 0$$

[3] 厳密な議論は陰関数定理による。

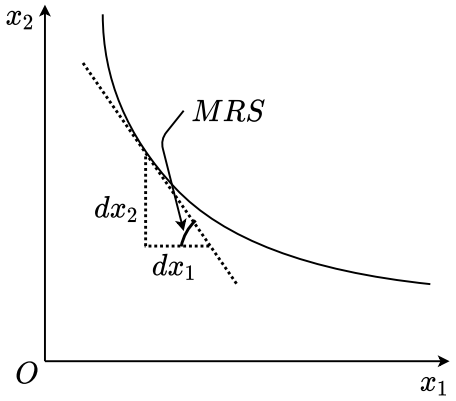


図 7.2: 限界代替率

7.2.2 予算制約

消費者は予算に限りがあるので，無尽蔵に消費を増やすわけにはいかない。財 1, 2 の価格をそれぞれ $p_1, p_2 > 0$, 消費者の予算を $m > 0$ としよう。すると，消費者は

$$p_1 x_1 + p_2 x_2 \leq m$$

が成り立つように x_1, x_2 を選ばなければならない。この不等式が成り立つ (x_1, x_2) の集合を「予算集合」（budget set）という。この不等式が等号で成り立つ集合（線分）を「予算線」（budget line）という。予算集合の不等式は

$$\frac{x_1}{m/p_1} + \frac{x_2}{m/p_2} \leq 1$$

と書けるので，(x_1, x_2) 平面に予算集合を描くと，横軸切片が　m/p_1, 縦軸切片が m/p_2 であるような線分（傾きは $-p_1/p_2$）と，横軸，縦軸で囲まれた直角三角形の領域になる。図 7.1 の影付きの直角三角形の領域が予算集合で，斜辺が予算線である。

7.2.3 効用最大化と均衡

消費者は，予算制約を満たすという条件の下で，効用 $U(x_1, x_2)$ が最大になるような (x_1, x_2) を選ぶ，という問題に直面している。これを数学的な表現を用いて次のように書く。

$$\max_{x_1,x_2} U(x_1,x_2)$$
$$\text{subject to} \qquad (P1)$$
$$p_1x_1 + p_2x_2 \le m$$

ここで，p_1, p_2, m は定数パラメータと考える。

効用を最大化する (x_1, x_2) を見つけることを「効用最大化問題を解く」，より一般に「最適化問題を解く」という。また，効用（より一般には「目的関数」）を最大化すると判明した (x_1, x_2) を「解」と呼ぶ。解を (x_1^*, x_2^*) と表現すると，

$$p_1x_1^* + p_2x_2^* \le m$$

と

$$p_1x_1 + p_2x_2 \le m \Longrightarrow U(x_1^*, x_2^*) \ge U(x_1, x_2)$$

が同時に成り立っている。つまり，

- 解は予算制約を満たす，
- 解は予算制約を満たす他の選択肢の中で，目的関数の値が最大である。

上記の効用最大化問題を解くためには，難しい数学は必要ない。次の 2 つのことがわかっていればよい [4]。

- 予算を余らせても効用に影響しないので，予算は使い切る方がよい。つまり，

$$p_1x_1 + p_2x_2 = m$$

- 無差別曲線はできるだけ右上にあるものの方がよいので，ギリギリ予算線に接するような無差別曲線を選ぶのがよい。つまり，無差別曲線の接線の傾き（MRS）が予算線の傾きと一致する。

$$MRS = -\frac{\partial U/\partial x_1}{\partial U/\partial x_2} = -\frac{p_1}{p_2} \tag{7.1}$$

図 7.1 の E 点が効用を最大化する消費点である。I 点は効用は高いものの，予算集合外にあるので解にならない。F 点は限界代替率と相対価格に関する条件 (7.1) を満たしていないので，解にならない。

条件 (7.1)

$$\frac{\partial U/\partial x_1}{\partial U/\partial x_2} = \frac{p_1}{p_2} \tag{7.2}$$

[4] 無差別曲線の形状や配置に仮定したことに強く依存しているが，技術的になりすぎるので説明は省略する。

は特に重要なので経済的な意味を確認しておこう。右辺は財 1 の財 2 に対する相対価格であり，2 財に対する市場の評価を表している。財 1 を 1 単位多く消費するためには，財 2 は p_1/p_2 単位だけ減らさなければならない。

$$p_1 \times 1 + p_2 \times \left(-\frac{p_1}{p_2}\right) = 0$$

左辺の方はすでに説明したとおり，効用を変化させない代替の比率である。これは 2 財に対する消費者の主観的な評価を表している。式 (7.2) が成り立っていないときには，消費の配分をかえることで効用を高めることができる。このとき，最大化は達成されていない。

「(7.2) が成り立たないなら最大ではない」からと言って,「(7.2) が成り立つときに最大である」とは言えないことに注意しよう。(7.2) は最大化問題の解が満たすべき必要条件であるが，十分条件であるとは限らない。しかし，われわれは無差別曲線に多くのことを仮定したので（原点に向かって厳密に凸な形状，右上の方が効用が高い），この問題はクリアされている。不安な読者はミクロ経済学か数理経済学の教科書を参考にするとよい。

問題 7.1. 予算線上にある消費点 (x_1, x_2) では，

$$\frac{\partial U/\partial x_1}{\partial U/\partial x_2} > \frac{p_1}{p_2}$$

である。このとき，x_1, x_2 をどのように変化させれば効用を高められるか。理由とともに答えなさい。（ヒント：図 7.1 の点 F から点 E に移動させることが必要である）

7.2.4 ラグランジュ未定乗数法

前述のような最適化問題を機械的に解く方法としてラグランジュ未定乗数法というものがあるので，簡単に解説しておこう。

$$\max_{x_1, x_2} U(x_1, x_2)$$

subject to

$$p_1 x_1 + p_2 x_2 \leq m$$

ラグランジアンと呼ばれる関数を次のように定義する。

$$\mathcal{L} = \underbrace{U(x_1, x_2)}_{\text{目的関数}} + \lambda \underbrace{\left[m - \left(p_1 x_1 + p_2 x_2\right)\right]}_{\text{制約条件, } \geq 0}$$

確実に $p_1 x_1 + p_2 x_2 = m$ となることを保証する条件が与えられている場合には，ラグランジアン $\mathcal{L}$ を x_1, x_2 と λ （未定乗数）で偏微分してゼロとなる点が解になる。

$$\frac{\partial \mathcal{L}}{\partial x_1} = \frac{\partial U}{\partial x_1} - \lambda p_1 = 0 \tag{7.3}$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = \frac{\partial U}{\partial x_2} - \lambda p_2 = 0 \tag{7.4}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = m - \left(p_1 x_1 + p_2 x_2\right) \tag{7.5}$$

もし，$p_1 x_1 + p_2 x_2 < m$ となる可能性がある場合には，(7.5) の代わりに

$$\lambda \left[m - \left(p_1 x_1 + p_2 x_2\right)\right] = 0$$

という条件を使う。制約条件にゆとり（スラック）がある場合には λ がゼロになる。

例 7.1. 効用関数をコブダグラス型

$$U(x_1, x_2) = x_1^{\alpha} x_2^{1-\alpha} \tag{7.6}$$

に特定化しよう。このとき，

$$\mathcal{L} = x_1^{\alpha} x_2^{1-\alpha} + \lambda \left[m - \left(p_1 x_1 + p_2 x_2\right)\right]$$

$$\frac{\partial \mathcal{L}}{\partial x_1} = \alpha x_1^{\alpha-1} x_2^{1-\alpha} - \lambda p_1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial x_2} = (1-\alpha) x_1^{\alpha} x_2^{-\alpha} - \lambda p_2 = 0$$

より，

$$\frac{\alpha}{1-\alpha}\frac{x_2}{x_1} = \frac{p_1}{p_2} \Longrightarrow \alpha p_2 x_2 = (1-\alpha) p_1 x_1 \tag{7.7}$$

を得る。さらに，

$$\frac{\partial \mathcal{L}}{\partial \lambda} = m - \left(p_1 x_1 + p_2 x_2\right) = 0$$

(7.7) を使うと，

$$x_1 = \frac{\alpha m}{p_1}, \qquad x_2 = \frac{(1-\alpha) m}{p_2} \tag{7.8}$$

を得る。これが財 1, 2 の需要関数になる。

未定乗数についても解いておこう。

$$\lambda = \frac{\alpha x_1^{\alpha-1} x_2^{1-\alpha}}{p_1} = \frac{\alpha}{p_1}\left(\frac{x_2}{x_1}\right)^{1-\alpha} = \frac{\alpha}{p_1}\left(\frac{(1-\alpha)p_1}{\alpha p_2}\right)^{1-\alpha} = \left(\frac{\alpha}{p_1}\right)^{\alpha}\left(\frac{1-\alpha}{p_2}\right)^{1-\alpha}$$

式 (7.6) と (7.8) を考慮して λ を解釈できる。λ は所得 m を 1 単位増やしたときに得られる追加的な効用（限界効用）である。追加的な 1 単位の所得は λ の分だけ効用を増加させる。

例 7.2. 効用関数を準線形と仮定しよう。

$$U(x_1, x_2) = \log x_1 + x_2$$

ただし，このような関数形の場合には $x_1 = 0$ や $x_2 = 0$ となる可能性がある（端点解）。こういうときには，$x_1 \geq 0, x_2 \geq 0$ という暗黙の制約条件をきちんと考慮してやる必要がある。ラグランジアンは

$$\mathcal{L} = u(x_1) + x_2 + \lambda\left[m - p_1 x_1 - p_2 x_2\right] + \lambda_1 x_1 + \lambda_2 x_2$$

になる。解が満たす条件は,

$$\begin{aligned}
\frac{1}{x_1} - \lambda p_1 + \lambda_1 &= 0 \\
1 - \lambda p_2 + \lambda_2 &= 0 \\
p_1 x_1 + p_2 x_2 &= m \\
\lambda_1 x_1 &= 0 \\
\lambda_2 x_2 &= 0
\end{aligned}$$

となる。少し整理すると,

$$\begin{aligned}
p_1 x_1 + p_2 x_2 &= m \\
\lambda p_1 x_1 &= 1 \\
\left[\lambda p_2 - 1\right] x_2 &= 0
\end{aligned}$$

まず，$x_2 \geq 0$ であると仮定してみよう。このとき,

$$\lambda = \frac{1}{p_2}, \qquad x_1 = \frac{p_2}{p_1}, \qquad x_2 = \frac{m - p_2}{p_2}$$

となる。$x_2 \geq 0$ であるためには，$m \geq p_2$ でなければならないようだ。

では，$p_2 < m$ のときどうなるか。このとき，$x_2 = 0$ となるので，$\lambda p_2 - 1$ は不定である。しかし，$p_1 x_1 + p_2 x_2 = m$ から

$$x_1 = \frac{m}{p_1}$$

と分かる。

7.3 理論: 2 期間の最適消費モデル

前節では，静学的な設定のもとで，2 財の選択に関する意思決定の性質を説明した。この節では，同じ財の 2 期間にわたる消費支出の配分を考える。動学的な計画問題であっても，均衡決定に関する考え方は同じであることを示す。しかし，時間経過には自然な順序関係があるので，分析に追加的な視点が加わることになる。例えば，消費量の経時的な増減を観察することで，経済成長についての含意を引き出せるかもしれない。

まずは，将来と未来をつなぐ「割引」について説明する。その後，2 期間の最適消費モデルを定式化し，均衡消費を求める。

7.3.1 価値の割引

1 年後に 100 万円を受け取る契約を考えよう。この契約の，現時点における価値はいくらだろうか。100 万円より小さいということが直感的にも明らかだろう。1 年後に 100 万円受け取る権利をわざわざ 100 万円払って購入する人はいないだろう [5]。額面上の同額を翌年にスライドさせるよりも，利子の付く金融資産で 100 万円を運用する方が得だからだ。話を簡単にするために，マクロ経済には唯一の名目利子率があると仮定しよう [6]。名目利子率は貨幣価値を翌年に持ち越すときの額面評価額の拡大率のことである。今の 100 万円は 1 年後の $100(1+i)$ 万円と等価である。逆に言えば，来年の 100 万円は今の $100/(1+i)$ 万円と等価になる。将来の貨幣価値を現在の貨幣価値に換算する公式は

$$\text{現在の貨幣価値} = \frac{\text{将来の貨幣価値}}{1 + \text{名目利子率}}$$

であり，この操作を**割引**という。

貨幣価値はインフレ率の変化によって上下するので，名目利子率からインフレ率を除去した実質利子率を考えるのが便利である。p_t を t 期の価格，x_t を数量，i_t を名目利子率とする。これらはスカラー（ベクトルではない普通の数）である。貨幣価値は $p_t x_t$ のよ

[5] 犯罪多発などの理由でタンス預金が現実的でなければ，そのような契約もあるかもしれない。

[6] 実際には，借り手の信用力やマクロ経済環境に応じて各個人が支払わなければならない利子率は異なる。貸すときの利子率（預金につく利子率）は借りるときの利子率（ローンに課される利子率）よりもはるかに小さい。したがって，利子率が 1 つだけというのは現実には観測されないが，単純化の仮定として広く受け入れられている。

うに書けるので，上記の割引の公式を置き換えて，

$$p_{t-1}x_{t-1} = \frac{p_t x_t}{1+i_t}$$

とできる。

$$x_{t-1} = \frac{x_t}{(1+i_t)\left(\frac{p_{t-1}}{p_t}\right)}$$

インフレ率を $\pi_t = (p_t/p_{t-1}) - 1$ とすれば，分母を次のように変形できる。

$$(1+i_t)\left(\frac{p_{t-1}}{p_t}\right) = \frac{1+i_t}{1+\pi_t} = 1 + r_t$$

このように定義した r_t を用いると，実質的な価値の割引の公式を得る。

$$x_{t-1} = \frac{x_t}{1+r_t}$$

r_t は実質利子率と呼ばれる。

注意 7.1. 実質利子率の定義を近似的に，

$$r = \frac{1+i}{1+\pi} - 1 \approx i - \pi$$

とすることができる。定義式両辺の対数を取ると

$$\log(1+i) = \log(1+r) + \log(1+\pi)$$

となる。ここでも再び瞬時成長率と離散時間の実効成長率の差が現れる。瞬時成長率については，

$$i = r + \pi$$

が厳密な等号で成立している。名目利子率，実質利子率，インフレ率の関係を表すこの公式は**フィッシャー方程式**と呼ばれている。

7.3.2 効用の割引

今年の消費から得られる効用と来年同じだけ消費することで得られる効用とを，今年のあなたが比較するとしよう。たとえば，今，寿司を食べに行くのと，1 年後に寿司を食べに行くことを計画するのでは，どちらが現在の効用を高めるだろうか。同じ消費行動をするのであれば，現在の消費と較べて将来の消費の方が効用に与える影響は小さくなるはずだ。価値と同様に，効用についても割り引く必要がある。

このような考察をもとに，次のような効用関数を使用することが考えられる。

$$U(c_1, c_2) = u(c_1) + \frac{u(c_2)}{1+\rho} = u(c_1) + \beta u(c_2)$$

$u(\cdot)$ を期間効用関数と呼ぶ。毎期の消費を当期の視点で評価したものである。将来の消費から得られる効用が現在の消費と比べて小さくなることを $1/(1+\rho)$ で掛けることで表現する。$\rho > 0$ を割引率，$1/(1+\rho) = \beta < 1$ を割引因子と呼ぶ。

7.3.3 予算制約

ある消費者が効用

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

を最大化するように消費計画 c_1, c_2 を選びたいとしよう。c_1 は 1 期の実質消費，c_2 は 2 期の実質消費である。この消費者はどのような予算制約に直面しているだろうか。

この消費者は 1 期，2 期のそれぞれにおいて y_1, y_2 の実質労働所得を稼ぐとしよう。この経済では，期首に保有している貯蓄残高に対して利払いを受けられる。例えば，消費者の計画時点での貯蓄残高 s_0 に対して，利子 $r_1 s_0$ が 1 期に支払われる。1 期の期末にかけての貯蓄残高の変化 $s_1 - s_0$ は，1 期に受け取る労働所得 y_1 と利子所得 $r_1 s_0$ および 1 期に支払う消費支出 c_1 を用いて，次のように表現できる。

$$s_1 - s_0 = y_1 + r_1 s_0 - c_1$$

あるいは

$$c_1 = y_1 + (1 + r_1)s_0 - s_1$$

同様にして 2 期のお金の出入りは次のように表現できる。

$$c_2 = y_2 + (1 + r_2)s_1 - s_2$$

2 期の式を 1 期分割り引くと，

$$c_1 = y_1 + (1 + r_1)s_0 - s_1$$
$$\frac{c_2}{1+r_2} = \frac{y_2}{1+r_2} + s_1 - \frac{1}{1+r_2}s_2$$

辺々足すと s_1 が消えて，

$$c_1 + \frac{c_2}{1+r_2} + \frac{s_2}{1+r_2} = (1 + r_1)s_0 + y_1 + \frac{y_2}{1+r_2}$$

を得る。

貯蓄残高はマイナスにもなることに注意しよう。負の残高は債務を表している。この消

費は c_1, c_2, s_2 を上手に選んで $U(c_1, c_2)$ を最大にしたい。しかし，s_2 がいくらでも大きなマイナスの値になるのであれば，効用関数は際限なく大きくできるだろう。最大化問題は意味をなさなくなってしまう。s_2 には下限を設定する必要がある。簡単のために，借金を残して死ぬ計画を立てないという条件を課そう。すなわち，

$$s_2 \geq 0$$

である。この条件の下では，

$$c_1 + \frac{c_2}{1+r_2} \leq (1+r_1)s_0 + y_1 + \frac{y_2}{1+r_2}$$

となる。表現を簡単にするため，$r_1 = 1, r_2 = r$ と書き直そう。考慮すべき制約条件は次のようになる。

$$c_1 + \frac{c_2}{1+r} \leq s_0 + y_1 + \frac{y_2}{1+r} \tag{7.9}$$

この式は

消費の割引現在価値の総和 ≤ 所得の割引現在価値の総和

という形式になっている。生涯所得以上には消費することはできないというまっとうな条件になっている。

なお，生涯所得に関する制約が成り立っている限りは 1 期末の借入れをいくらでも大きくすることができることに注意しよう。つまり，$s_2 \geq 0$ ではあるものの，s_1 については非常に大きなマイナスになっても構わない。この条件については，

$$s_1 \geq 0 \tag{7.10}$$

あるいは

$$s_1 \geq -\underline{s}$$

のような条件を付加して修正することがある。このような追加的な制約条件を借入制約（borrowing constraint）と呼ぶ。

問題 7.2. (7.9) を満たす c_1, c_2 の範囲を (c_1, c_2) 平面に描きなさい。

問題 7.3. (7.9) と (7.10) を満たす c_1, c_2 の範囲を (c_1, c_2) 平面に描きなさい。

7.3.4 解法

借入制約のないケースを考えよう。消費者は

$$\begin{aligned} &\max u(c_1) + \beta u(c_2) \\ &\text{subject to} \qquad (P2) \\ &\quad c_1 + \frac{c_2}{1+r} \leq s_0 + y_1 + \frac{y_2}{1+r} \end{aligned}$$

を解く。ここで，この問題を解く際には s_0, y_1, y_2, r, β は定数パラメータと考える。

c_1 の価格が 1, c_2 の価格が $1/(1+r)$ と解釈すれば，問題 $(P1)$ と $(P2)$ は同じ形式を持っていることが分かるだろう。したがって，限界代替率が相対価格と一致するという条件が解を特徴づける。

$$MRS = \frac{u'(c_1)}{\beta u'(c_2)}$$

$$\text{相対価格} = \frac{1}{1/(1+r)} = 1 + r$$

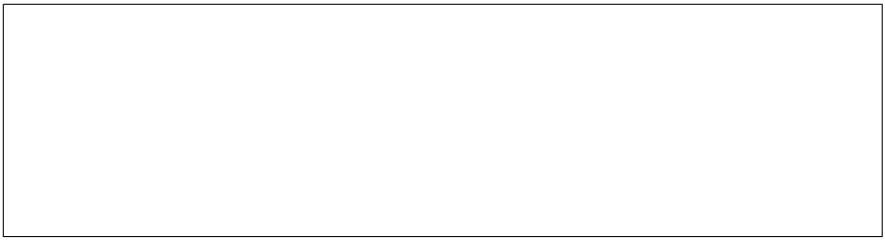
したがって，最適性の必要条件は

$$\frac{u'(c_1)}{u'(c_2)} = \beta(1+r) \tag{7.11}$$

$$c_1 + \frac{c_2}{1+r} = s_0 + y_1 + \frac{y_2}{1+r}$$

となる。

問題 7.4. ラグランジュ未定乗数法を用いて同じ問題を解きなさい。また，ラグランジュ乗数 λ の経済的な意味を説明しなさい。



7.3.5 解の性質

一般的な効用関数では解釈性が難しいので期間効用を特定化する。特に扱いが容易なのは，相対的危険回避

$$\Theta(c) = -\frac{cu''(c)}{u'(c)}$$

が一定であるような効用関数である。これを CRRA 型効用関数（Constant Relative Risk Aversion）と呼ぶ。

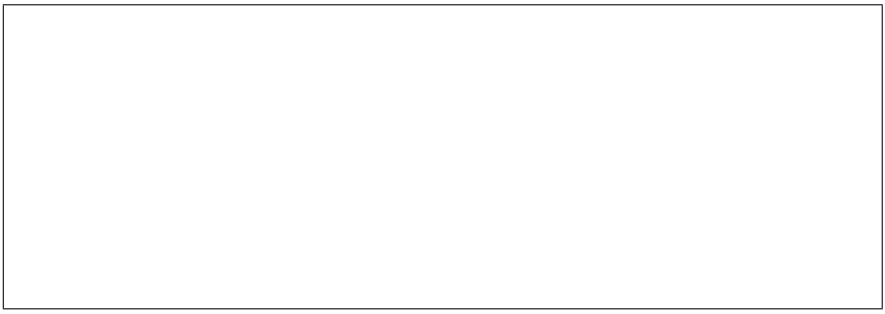
$$u(c) = \begin{cases} \frac{c^{1-\theta}-1}{1-\theta} & \text{if} \quad \theta \neq 1 \\ \log c & \text{if} \quad \theta = 1 \end{cases}$$

限界効用逓減を要請するので，$\theta > 0$ でなければならない [7]。

問題 7.5. CRRA 型効用関数に対して，

$$\Theta(c) = \theta$$

が成り立つことを確認しなさい。



[7] 期間効用関数の限界効用が逓減しないと，総効用 $U(c_1, c_2)$ が原点に向かって厳密に凸にならない。無差別曲線が直線的な形状（ファセットと呼ぶ）を持つと，解が無数に存在する，端点解が生じるといった厄介な問題が生じてくる。

CRRA 型を仮定すると，(7.11) の条件は

$$\left(\frac{c_2}{c_1}\right)^{\theta} = \beta(1+r) = \frac{1+r}{1+\rho}$$

とできる。

$$\frac{c_2}{c_1} = \left(\frac{1+r}{1+\rho}\right)^{1/\theta} \tag{7.12}$$

$\theta > 0$ より，次の命題を得る。

命題 7.1. *2 期間最適消費問題 $(P2)$ の最適解 (c_1^*, c_2^*) は*

$$r \lessgtr \rho \Leftrightarrow c_2^* \lessgtr c_1^*$$

を満たす。

利子率 r は消費を遅らせることで追加的に得られる所得である。割引率 ρ は消費を遅らせることに対する効用のペナルティである。待つことの便益が相対的に小さいとき（$r < \rho$）には早めにたくさん消費するだろうから，$c_1^* > c_2^*$ が成り立つのである。

1 期の貯蓄 s_1 について見ておこう。毎期のお金の出入の式に $s_2 = 0$ を代入すると以下の式を得る。

$$\begin{aligned} c_1 &= y_1 + s_0 - s_1 \\ c_2 &= y_2 + (1+r)s_1 \end{aligned}$$

ここで，(7.12) を使うと

$$\frac{y_2 + (1+r)s_1}{y_1 + s_0 - s_1} = \left(\frac{1+r}{1+\rho}\right)^{1/\theta}$$

これを s_1 について解く。

$$\begin{aligned} s_1 &= \frac{\left(\frac{1+r}{1+\rho}\right)^{1/\theta}(y_1 + s_0) - y_2}{(1+r) + \left(\frac{1+r}{1+\rho}\right)^{1/\theta}} \\ &= \frac{(1+r)^{1/\theta}(y_1 + s_0) - (1+\rho)^{1/\theta} y_2}{(1+r)(1+\rho)^{1/\theta} + (1+r)^{1/\theta}} \end{aligned}$$

消費者の効用最大化問題によって消費・貯蓄行動を特徴づけると，ソロー・モデルが想定するような単純な貯蓄関数と比べてはるかに複雑な貯蓄関数になる。

例 7.3. $y_2 = 0, s_0 = 0$ のケースは貯蓄関数についてもう少し考察をすすめることができる。この仮定は 2 期間世代重複モデルでも使われる。

$$s_1 = \left[\frac{(1+r)^{\frac{1-\theta}{\theta}}}{(1+\rho)^{\frac{1}{\theta}} + (1+r)^{\frac{1-\theta}{\theta}}} \right] y_1 \tag{7.13}$$

当期の所得の一部が貯蓄に割り振られるという式になっている。ただし，利子率 r に依存していることに注意しよう。ソロー・モデルの分析で学んだように，利子率は資本の限界生産性 MPK で決まるので，完全な定率という訳ではない。しかし，さらに $\theta = 1$ という条件をつければ利子率への依存性がなくなる。

$$s_1 = \frac{1}{2+\rho} y_1$$

利子率の上昇は消費の意思決定において 2 つの効果を引き起こす。1 つは第 2 期の財価格（$1/(1+r)$）が下落することで，第 1 期の消費から第 2 期の消費への代替が進む。つまり第 1 期の消費を減少させる作用が働く。これを**代替効果**（substitution effect）という[8]。もう 1 つは**所得効果**（income effect）である。価格の下落により購買力が増加することで，第 1 期，第 2 期ともに消費を増加させる作用が働く。利子率 r の上昇に伴って貯蓄が増加する場合，第 1 期の消費が減少しているはずなので，代替効果が所得効果を上回っていることになる。逆に，利子率の上昇に伴って貯蓄が減少する場合，第 1 期の消費が増加しているので，所得効果が代替効果を上回っている。$\theta = 1$ のときには所得効果と代替効果が相殺し合う。

問題 7.6. (7.13) が r の増加関数になるのは $\theta > 1$ のときか，$\theta < 1$ のときか。判定せよ。

7.4 プログラミング

この章のプログラミングパートでは，Python のシンボリック計算ライブラリ **SymPy** と数値最適化のための scipy.optimize モジュール（**SciPy** ライブラリの一部）を紹介する。

[8]代替効果には，スルツキーの代替効果とヒックスの代替効果がある。スルツキーの代替効果を求めるときには，価格の変化が起こる前の最適消費点が価格変化後の予算線に乗るように所得の調整を行ったあとで，新しい予算線上に最適消費点を探す。スルツキーの代替効果は効用関数の形状によらずに必ず非正になる。無限小の価格変化に関するスルツキー分解では 2 つの代替効果は一致する。

7.4.1 シンボリック計算と数値計算

コンピュータを用いた数学的な計算には，シンボリックな計算と数値的な計算がある。シンボリックな計算というのは，記号の操作のみを用いて計算を行うことである。例えば，四則演算とべきからなる有限の操作だけを用いて計算をする代数計算や，初等関数の微分，一部の不定積分の計算などはシンボリックな計算の例である。数値的な計算というのは，具体的な数値と近似を用いて答えを導く操作と考えておけばよい。

シンボリック計算の身近な例としては，2 次方程式

$$ax^2 + bx + c = 0$$

の求解問題がある。 $a \neq 0$ のとき，

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

であることはよく知られている。係数の四則演算とべき根からなる有限回の操作によって解が表現されている。このような求解の手続きを**代数的に解く**と呼ぶ。得られた解表現を**解の公式**という。4 次方程式までは解の公式が存在することが知られている。5 次以上の一般の代数方程式（多項式 = 0 なる方程式）には解の公式が存在しない [9]。

5 次以上の方程式では解を見つけることはできないかというと，そうではない。解 x^* に収束する数列 $\{x_n\}$ をうまく見つけられれば（$\lim_{n\to\infty} x_n = x^*$），十分大きな N を用いて $x_N \approx x^*$ という近似式が成り立つ。$\lim_{n\to\infty} x_n = x^*$ が成り立つことを証明する手続きを**解析的に解く**という。解析的な解の近似によって解を求めることを**数値的に解く**，という [10]。

7.4.2 シンボリック計算

Python で代数計算（シンボリック計算）を行うときは **SymPy** というライブラリを用いる。このセクションのコードは Jupyter Notebook を用いて実行することを推奨する。次のコードを実行してインポートと，出力される数式の見栄えを調整する。

```
import sympy as sp
sp.init_printing()
```

まずはシンボルとして用いる変数を定義する。アルファベットの単純なシンボルを定義するときには次のように `from sympy.abc import ...` というイディオムを使うとよい。

[9] アーベル=ルフィニの定理。「一般」という表現を誤解しないように。これは，「解の公式が存在しない代数方程式がある」ということ。解の公式が存在する部分集合の存在は否定していない。

[10] 解析的に解くという表現は必ずしも数値的に解くことと同義ではない。数列の各ステップ x_n の計算に無限の自由度を伴う変数（関数）の操作が必要な場合には数値的に解けない。コンピュータの物理的な制約で無限状態を表現できないからだ。したがって，あらゆる計算を有限近似で表現できたときにはじめて数値的に解くアルゴリズムを得たことになる。

```
from sympy.abc import x, y, a, b, c, f
```

シンボリックな計算とは，次のような計算のことである。あまりにも自明の計算ではあるが，コンピュータにも私たちが行うような普通の，抽象的な計算ができるところに驚きがある。

```
x ** 2 *  y / x
```

$$xy$$

2 次方程式も次のように解いてくれる。

```
sp.solve(a * x**2 + b * x + c, x)
```

$$\left[\frac{-b+\sqrt{-4ac+b^2}}{2a}, -\frac{b+\sqrt{-4ac+b^2}}{2a}\right]$$

シンボリック計算は代数計算だけではなく，微分の計算なども含まれる。公式の適用で機械的にできる操作はシンボリックな計算ができるのだ。

```
f = x**a * y**b
f.diff(x)
```

$$\frac{ax^a y^b}{x}$$

これは，なんとなくおかしい気がする。分子と分母の x をまとめてくれたりはしないようだ。こういうときには simplify() というメソッドを使うとうまく整理してくれることがある。ただし，いつもうまくいくとは限らないことに注意しよう。

```
f.diff(x).simplify()
```

$$ax^{a-1}y^b$$

CRRA 型効用関数

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

の相対的危険回避度 $-cu''/u'$ が定数であることを確認してみよう。ギリシャ文字は英語表記を使うとうまく表示してくれる。

```
c, theta = sp.symbols("c theta")
u = (c**(1 - theta) - 1) / (1 - theta)
u
```

$$\frac{c^{1-\theta}-1}{1-\theta}$$

```
(- c * u.diff(c, 2) / u.diff(c)).simplify()
```

$$\theta$$

シンボリックな関数に数値を代入したいときがある。例えば，ソロー・モデルの定常状態を考えてみよう。コブ=ダグラス型 $f(k) = k^{\alpha}$ を仮定すると，定常状態の効率労働あたり資本 k^* が満たす方程式は

$$sk^{\alpha} - (\delta + g + n + gn)k = 0$$

であり，**SymPy** では次のように求める。解きたい方程式を (式) = 0 の形にして，左辺の式のみを指定していることに注意しよう。

```
k, s, delta, alpha, g, n = sp.symbols("k s delta alpha g n")
ss = sp.solve(s * k**alpha - (delta + g + n + g * n) * k, k)
ss
```

$$\left[\left(\frac{\delta + gn + g + n}{s}\right)^{\frac{1}{\alpha-1}}\right]$$

ここで，パラメータ α, s, δ, g, n を代入した値を計算したければ，次のように subs() メソッドを使うとよい。入力は辞書である。キーとしてパラメータのシンボル，値には代入したいパラメータ値を設定する。ss[0] としているのはリストの第 0 要素を抽出するお馴染みのコードだ。一般の方程式には複数個の解があるので，sp.solve() の結果は必ずリストになる。

```
ss[0].subs({alpha: 0.3, s: 0.4, delta: 0.02, g: 0.03, n: 0.02})
```

```
11.9145868157865
```

すべてのパラメータに数値を代入するのではなく，例えば，パラメータ s を変化させたときに，定常状態の変化を見たいとしよう。s だけは代入せずにおいておく。

```
ks = ss[0].subs({alpha: 0.3, delta: 0.02, g: 0.03, n: 0.02})
ks
```

$$\frac{44.1128089260847}{\left(\frac{1}{s}\right)^{1.42857142857143}}$$

このオブジェクトを s の関数として扱いたいなら，**SymPy** のシンボリックなオブジェクトから数値的な関数を生成する必要がある [11]。これには sp.lambdify() という関数を使う。シンボリック計算から数値計算に橋渡しをしてくれる重要な関数である。

[11] 作図するだけなら sympy.plotting モジュールも使うことができる。https://docs.sympy.org/latest/modules/plotting.html

```
f_ks = sp.lambdify(s, ks)
f_ks(0.3)
```

```
7.899417661971007
```

プロットしてみよう。

```
import numpy as np
import matplotlib.pyplot as plt

sgrid = np.linspace(0.001, 1, 100)
plt.plot(sgrid, f_ks(sgrid))
plt.xlabel("s")
plt.ylabel("k*")
plt.show()
```

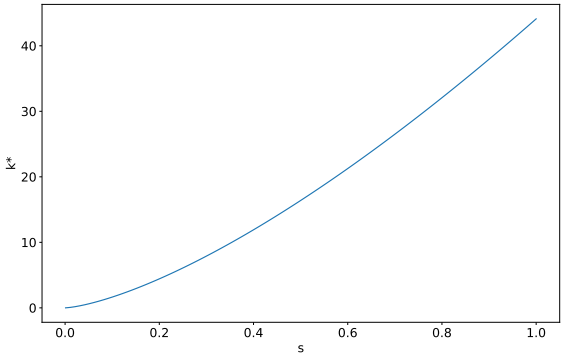


図 7.3: s と k^* の関係

例 7.4 (ラグランジュ未定乗数法)**.** 次の最大化問題を解いてみよう。

$$\max x^{1/2} y^{1/2}$$

subject to

$$x + 3y = 100$$

ラグランジアン

$$\mathcal{L} = x^{1/2}y^{1/2} + \lambda(100 - x - 3y)$$

として,

$$\frac{\partial \mathcal{L}}{\partial x} = 0, \quad \frac{\partial \mathcal{L}}{\partial y} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

となる (x, y, λ) を求めればよい [12]。

```
lamda = sp.symbols("lamda")
L = sp.sqrt(x * y) + lamda * (100 - x - 3 * y)
L
```

$$\lambda\left(-x - 3y + 100\right) + \sqrt{xy}$$

```
sp.solve([L.diff(x), L.diff(y), L.diff(lamda)])
```

$$\left[\left\{\lambda : \frac{\sqrt{3}}{6},\ x : 50,\ y : \frac{50}{3}\right\}\right]$$

うまくいった。しかし，この計算はいつでもうまくいくとは限らないことに注意しよう。シンボリックな計算では解けない方程式がある。

問題 7.7. 例 7.4 の数字を少し変えて，上記の手続きで解ける問題と解けない問題を 1 つずつ探しなさい。なお，**SymPy** では有理数は `sp.Rational(1/3)` などのようにして定義できる。

例 7.5 (多項式近似)**.** シンボリックな計算が適用できる状況は限られているが，私たちがモデル構築に用いるような普通の関数の微分計算は安全に行える。例えば,

$$f(x) = \sin 3x + e^{x/2}$$

を $x = x_0$ のまわりで，2 次方程式で近似する問題を考えよう。

[12] λ の英語表記は lambda だが，Python のキーワードなので使えないため lamda としている。Unicode の文字名としては lamda が正しいそうなので，あながち間違いという訳でもないようだが。

```
fx = sp.sin(3 * x) + sp.exp(x / 2)
fx
```

$$e^{\frac{x}{2}} + \sin\left(3x\right)$$

2 次までテイラー展開の公式は次の通りである。。

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2$$

テイラー近似の公式に忠実に従えば，次のように書ける。

```
fx_quad_withloop = (fx.subs({x: y})
                    + sum(fx.diff(x, i).subs({x: y}) * (x - y)**i / i
                          for i in range(1, 3)))
fx_quad_withloop
```

$$\frac{\left(x - y\right)^2\left(\frac{e^{\frac{y}{2}}}{4} - 9\sin\left(3y\right)\right)}{2} + \left(x - y\right)\left(\frac{e^{\frac{y}{2}}}{2} + 3\cos\left(3y\right)\right) + e^{\frac{y}{2}} + \sin\left(3y\right)$$

しかし，**SymPy** の `series()` メソッドを使うほうが簡単だろう。`removeO()` メソッドは近似誤差を無視するためのものだ。

```
fx_quad = fx.series(x, y, n=3).removeO()
fx_quad
```

$$\left(x - y\right)^2\left(\frac{\sqrt{e^y}}{8} - \frac{9\sin\left(3y\right)}{2}\right) + \left(x - y\right)\left(\frac{\sqrt{e^y}}{2} + 3\cos\left(3y\right)\right) + e^{\frac{y}{2}} + \sin\left(3y\right)$$

近似がうまくいっていることを目視で確かめよう。$x = 2$ の周りで近似し，$1 < x < 3$ の範囲でプロットしてみよう。結果は図 7.4 のようになる。

```
fx_fun = sp.lambdify(x, fx)
fx_quad_fun = sp.lambdify(x, fx_quad.subs({y: 2}))

xgrid = np.linspace(1, 3)
plt.plot(xgrid, fx_fun(xgrid), label='Original')
plt.plot(xgrid, fx_quad_fun(xgrid), label='Quadratic')
plt.legend()
plt.show()
```

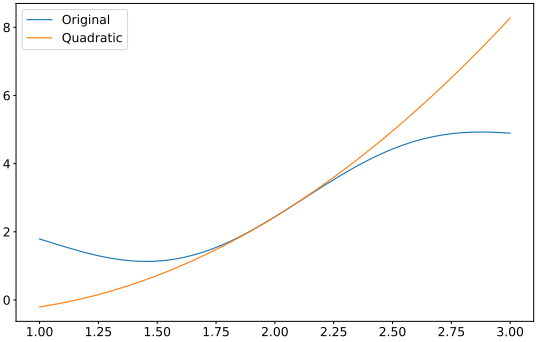


図 7.4: 2 次の多項式近似

7.4.3 数値計算

最適化問題における最も重要な課題は，ある関数がゼロになる点を探すことである。本書では経済モデルの解を可視化したり数値的に評価したりすることに主眼を置いているので，求解は代数的な方法ではなく数値的（近似的）な方法で十分である。

最も簡単な 1 変数の実数値関数を用いて数値求解の概要を説明しよう。図 7.5 の太実線で描かれた曲線が $f(x) = 0$ になる x を探したい関数であるとする。適当な初期値 x_0 を設定する。この点で $f(x)$ の接線を引く。接線の方程式は，

$$y = f(x_0) + f'(x_0)(x - x_0)$$

である。この接線の方程式の x 切片，つまり $y = 0$ となる x を探す。この点を x_1 と記すとすれば，

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$(x_1, f(x_1))$ で $f(x)$ の接線を引いて同じことをすると，

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

を定義できる。図 7.5 では徐々に $f(x_*) = 0$ なる x_* に近づいている。これをさらに繰り返して，

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \qquad n = 1, 2, \ldots$$

を定義すると，関数 f に対する適切な仮定のもとで

$$\lim_{n\to\infty} x_n = x_*$$

となることが示される。この数値求解法を**ニュートン・ラフソン法**と呼ぶ。

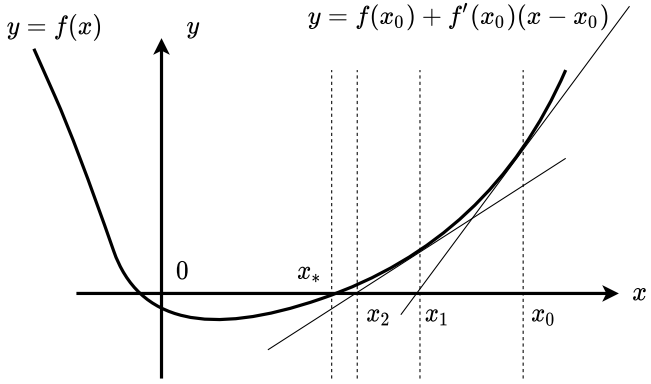


図 7.5: ニュートン・ラフソン法

本書はアルゴリズムの詳細な解説を目的としていないし，私では役者不足なので，これ以上解説を続けるのはやめておこう。数値球解法のユーザーとして必ず認識しておくべき教訓を述べるにとどめておく。

- 初期値を適切に選ばないと適切な解を見つけることができない。例えば，上で説明した手続きでは図 7.5 の原点の左にあるもう 1 つの解は見つけられない。逆に初期値をもっと左の方に取ると x_* を見つけられない。
- 実はニュートン・ラフソン法で求めた数列 $(x_n)_{n\geq 0}$ が収束しないケースがある。したがって，必ず収束性のチェックをしなければならない [13]。
- 初期値の設定と，収束性に関するチェックは他の求解アルゴリズムを用いる場合にも忘れてはいけない。

Python で数値的に求解したいときは **SciPy** ライブラリの optimize モジュールを用いる。`newton()` と `fsolve()` を紹介しよう。

```
from scipy.optimize import newton, fsolve
```

[13]Quora の質問「When does Newton Raphson fail?」に対する Alon Amit の解答を参照
https://www.quora.com/When-does-Newton-Raphson-fail（2020/07/03 閲覧）

newton()

1 変数関数 $f : \mathbb{R} \to \mathbb{R}$ のゼロ点を探すときには newton() 関数を使う [14]。引数はゼロを探したい関数 func，初期値 x0，導関数 fprime　である。簡単な例

$$f(x) = x^2 - 3, \qquad f'(x) = 2x$$

を用いて試してみよう。

```
def f(x):
    return x**2 - 3

def df(x):
    return 2 * x

newton(f, x0=3, fprime=df)
```

```
1.7320508075688772
```

初期値 x0 に **NumPy** 配列を渡すと複数の初期値から出発して複数の解を見つけられる。

```
newton(f, x0=np.array([-3, 3]), fprime=df)
```

```
[-1.73205081  1.73205081]
```

fsolve()

ベクトル値関数 $f : \mathbb{R}^n \to \mathbb{R}^n$ のゼロ点（ベクトルすべての要素がゼロになる点）を探すときには fsolve() 関数を使うとよい [15]。

異時点間の最適消費モデル

$$\begin{aligned} &\max \left[\log c_1 + \beta \log c_2\right] \\ &\text{subject to} \\ &\quad c_1 + \frac{c_2}{1+r} = 1000 \end{aligned}$$

を考えよう。$\beta = 0.95, r = 0.05$ とする。ラグランジアン

$$\mathcal{L} = \log c_1 + \beta \log c_2 + \lambda \left(1000 - c_1 - \frac{c_2}{1+r}\right)$$

[14] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html
[15] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html

を用いて,

$$\frac{\partial \mathcal{L}}{\partial c_1} = 0, \quad \frac{\partial \mathcal{L}}{\partial c_2} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

なる点 (c_1, c_2, λ) を探したい。最適のための 1 階条件は

$$\begin{aligned} \frac{1}{c_1} - \lambda &= 0 \\ \frac{\beta}{c_2} - \frac{\lambda}{1+r} &= 0 \\ 1000 - c_1 - \frac{c_2}{1+r} &= 0 \end{aligned}$$

である。したがって,

$$f(c_1, c_2, \lambda) = \begin{bmatrix} \frac{1}{c_1} - \lambda \\ \frac{\beta}{c_2} - \frac{\lambda}{1+r} \\ 1000 - c_1 - \frac{c_2}{1+r} \end{bmatrix} \tag{7.14}$$

がゼロになる点を探せばよい。

実はこの問題は数値計算をするまでもなく厳密に答えを見つけることができて，最適 $(c_1^*, c_2^*, \lambda^*)$ は

$$\begin{aligned} c_1^* &= \frac{1000}{1+\beta} \approx 512.821 \\ c_2^* &= \beta(1+r)c_1^* \approx 511.538 \\ \lambda^* &= 0.00195 \end{aligned}$$

である。

さて，数値的にも解いてみよう。ヤコビアン行列というベクトル関数の微分を計算して

おくと数値計算上都合がよい[16]。

$$Df(c_1, c_2, \lambda) = \begin{bmatrix} -\frac{1}{c_1^2} & 0 & -1 \\ 0 & -\frac{\beta}{c_2^2} & -\frac{1}{1+r} \\ -1 & -\frac{1}{1+r} & 0 \end{bmatrix} \tag{7.15}$$

準備が整ったので数値的に解いてみよう。下のコードの重要な部分を簡単に説明しておく。

- 関数 f は (7.14) で定義されたもの。
- 関数 df は (7.15) で定義されたヤコビアン行列。
- 関数 fsolve に渡している x0=[100, 100, 0.1] は (c_1, c_2, λ) の初期値である。

```
beta, r = 0.95, 0.05

def f(x):
    c1, c2, lam = x
    return [1 / c1 - lam,
            beta / c2 - lam / (1 + r),
            1000 - c1 - c2 / (1 + r)]

def df(x):
    c1, c2, lam = x
    return [[-1 / c1**2, 0,             -1],
            [0,          -beta / c2**2, -1 / (1 + r)],
            [-1,         -1 / (1 + r),  0]]

fsolve(f, x0=[100, 100, 0.1], fprime=df)
```

```
[512.82051282 511.53846154   0.00195    ]
```

[16]ベクトル関数

$$f(x_1, x_2, x_3) = \begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \\ f_3(x_1, x_2, x_3) \end{bmatrix}$$

に対して，ヤコビアン Df は次のように定義される。

$$Df(x_1, x_2, x_3) = \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \partial f_1/\partial x_3 \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \partial f_2/\partial x_3 \\ \partial f_3/\partial x_1 & \partial f_3/\partial x_2 & \partial f_3/\partial x_3 \end{bmatrix}$$

本文の最適化問題では f 自体もラグランジアン $\mathcal{L}$ の微分（勾配）によって定義される。これを $\nabla\mathcal{L}$ と書く，そのヤコビアンは $\mathcal{L}$ のヘッセ行列であり，$\nabla^2\mathcal{L}$ と書くことが多い。∇ は ナブラと読む。

7.4.4 シンボリック計算と数値計算のあわせ技

ようやく標準的なワークフローを説明する準備が整った。

Python を使って最適化問題を解く手順は次のようになる。**SymPy** と **SciPy** の使い訳を確認してほしい。

1. 最適化問題をセットアップする。（紙とペン）
2. ラグランジアン $\mathcal{L}$ を定義する。（紙とペン）
3. ラグランジアンの勾配 $f = \nabla\mathcal{L}$ と，そのヤコビアン $Df = \nabla^2\mathcal{L}$ を計算する。（**SymPy**）
4. パラメータを固定して，$f = \nabla\mathcal{L}$ と $Df = \nabla^2\mathcal{L}$ を数値関数に変換する。（**SymPy**）
5. 適当な初期値と $Df = \nabla^2\mathcal{L}$ を与えて，$f = \nabla\mathcal{L}$ がゼロになる点を見つける。（**SciPy**）
6. その後，必要な数値的な分析を行う。（**NumPy**, **Pandas**）

Step 1　再び，異時点間の最適消費モデルを考えよう。効用関数は一般の CRRA 型とする。

$$\max \left[\left(\frac{c_1^{1-\theta} - 1}{1-\theta} \right) + \beta \left(\frac{c_2^{1-\theta} - 1}{1-\theta} \right) \right]$$

subject to

$$c_1 + \frac{c_2}{1+r} = 1000$$

```
c1, c2, theta, r, beta, lamda = sp.symbols("c_1 c_2 theta r beta lamda")
U = c1**(1 - theta) / (1 - theta) + beta * c2**(1 - theta) / (1 - theta)
U
```

$$\frac{\beta c_2^{1-\theta}}{1-\theta} + \frac{c_1^{1-\theta}}{1-\theta}$$

Step 2　ラグランジアンを定義する。λ が正になるように制約条件の扱い方に気をつけよう。

$$\mathcal{L} = \left(\frac{c_1^{1-\theta} - 1}{1-\theta} \right) + \beta \left(\frac{c_2^{1-\theta} - 1}{1-\theta} \right) + \lambda \left(1000 - c_1 - \frac{c_2}{1+r} \right)$$

```
L = U + lamda * (1000 - c1 - c2 / (1 + r))
L
```

$$\frac{\beta c_2^{1-\theta}}{1-\theta} + \frac{c_1^{1-\theta}}{1-\theta} + \lambda \left(-c_1 - \frac{c_2}{r+1} + 1000 \right)$$

Step 3　微分 $\nabla\mathcal{L}$ と $\nabla^2\mathcal{L}$ を計算する。これは，

$$\nabla \mathcal{L} = 0$$

となる点を探したいのと，最適化ルーチーン $\nabla^2 \mathcal{L}$ を渡すため。

まずは，$f = \nabla \mathcal{L}$ の計算。sp.Matrix() で行列形式にしておくと見やすい。[L.diff(v) for v in (c1, c2, lamda)] は (c1, c2, lamda) に含まれる各シンボルで微分したものをリストにまとめるという意味。

```
f_sp = sp.Matrix([L.diff(v) for v in (c1, c2, lamda)])
f_sp
```

$$\begin{bmatrix} -\lambda + \frac{c_1^{1-\theta}}{c_1} \\ \frac{\beta c_2^{1-\theta}}{c_2} - \frac{\lambda}{r+1} \\ -c_1 - \frac{c_2}{r+1} + 1000 \end{bmatrix}$$

続いて $Df = \nabla^2 \mathcal{L}$ の計算。これも sp.Matrix(sp.BlockMatrix()) で行列形式にしておく。

```
Df_sp = sp.Matrix(sp.BlockMatrix([f_sp.diff(v) for v in (c1, c2, lamda)]))
Df_sp
```

$$\begin{bmatrix} \frac{c_1^{1-\theta}(1-\theta)}{c_1^2} - \frac{c_1^{1-\theta}}{c_1^2} & 0 & -1 \\ 0 & \frac{\beta c_2^{1-\theta}(1-\theta)}{c_2^2} - \frac{\beta c_2^{1-\theta}}{c_2^2} & -\frac{1}{r+1} \\ -1 & -\frac{1}{r+1} & 0 \end{bmatrix}$$

Step 4　続いてパラメータの代入と数値関数への変換を行う。$\beta = 0.9$, $\theta = 0.3$, $r = 0.1$ という値を代入している。f の方だけ np.squeeze() をかぶせているのは，shape が (3,) の配列を入出力にとる関数にしたいから。

```
params = {beta: 0.90, theta: 0.3, r: 0.1}
f = sp.lambdify([(c1, c2, lamda)], np.squeeze(f_sp.subs(params)))
Df = sp.lambdify([(c1, c2, lamda)], Df_sp.subs(params))
```

Step 5　最後に fsolve() で最適化問題を解く。

```
c1_star, c2_star, lam_star = fsolve(f, x0=[500, 500, 1], fprime=Df)
c1_star, c2_star, lam_star
```

```
\left( 532.1583718714962, \  514.6257909413544, \
    0.15212049514588077\right)
```

Step 6 どのような計算でもよいのだが，例えば消費の変化率を計算したいとしよう。次のようにすればよい。

```
(c2_star - c1_star) / c1_star
```

```
-0.032946171397216284
```

パラメータを変えて最適消費点の変化を見たいときには，Step 4 と Step 5 を繰り返してデータを作っていけばよい。

例えば，次のコードは $\theta = 0.3, 1, 5$ のそれぞれに対して，r と 消費の変化率 $(c_2^* - c_1^*)/c_1^*$ の関係を計算し，プロットする。θ が大きくなると，消費に変動がなくなってくることが分かる。θ は同じ水準の消費を続けたいという消費者の選好を表しているのである。

```
N = 30
rv = np.linspace(0.01, 0.2, N)
theta_v = np.array([0.3, 1.0, 5])

for thta in theta_v:
    change = np.empty(N)
    for i in range(N):
        # Step 4
        params = {beta: 0.90, theta: thta, r: rv[i]}
        f = sp.lambdify([(c1, c2, lamda)], np.squeeze(f_sp.subs(params)))
        Df = sp.lambdify([(c1, c2, lamda)], Df_sp.subs(params))

        # Step 5
        c1_star, c2_star, lam_star = fsolve(f, x0=[500, 500, 1], fprime=Df)

        # Step 6
        change[i] = (c2_star - c1_star) / c1_star

    plt.plot(rv, change, label=f"theta={thta}")

plt.legend()
plt.xlabel("r")
plt.ylabel("Growth rate of c")
plt.show()
```

なお，ここで考えている最適化問題は実質利子率が変わっても生涯所得には変化がないことに注意しよう。問題 $(P2)$ をきちんと解く代わりに少し簡単な問題を解いたのである。問題 $(P2)$ を完全な形で解くことは読者の練習問題にしよう。

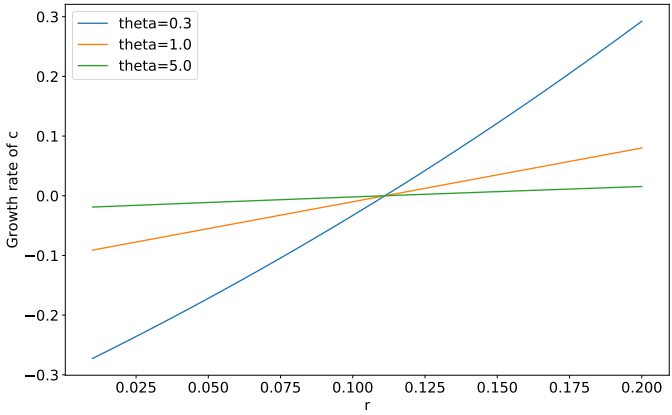


図 7.6: θ, r と消費の変化率の関係