第6章　連続最適化

6.1　目的

与えられた関数 $f\colon \mathbb{R}^n \to \mathbb{R}$ の零点，および最小値または最大値を与える点 $x \in \mathbb{R}^n$ を求める手法を学び，プログラミング言語 Python を用いた演習を通じて，その手法を理解する．内容は，6.2〜6.5 節の大きく 4 つに分かれており，6.2 節は関数の零点を求める二分法とニュートン法について学び，Python を導入する．6.3 節は降下法の概要を述べ，最適化手法として根幹を成す最急降下法およびニュートン法について学ぶ．6.4 節は，直線探索を用いたニュートン法および最適化手法の収束性について学ぶ．6.5 節は，ニュートン法に修正を加えた手法について学び，総合演習を行う．

6.2　関数の零点探索と Python 導入

ここでは，関数の零点探索のための二分法とニュートン法を学び，Python の導入を行う．

6.2.1　関数の零点探索

与えられた関数 $f\colon \mathbb{R} \to \mathbb{R}$ に対して，$f(x) = 0$ を満たす点 x のことを関数 f の零点と呼ぶ．関数の零点探索は，様々な場面で必要とされる．例えば，行列 $A \in \mathbb{R}^{n\times n}$ の固有値を求める際には，固有多項式 $p_A(x) = \det(xI - A)$ の零点を求めることに帰着され，一方，連続関数 $F\colon \mathbb{R} \to \mathbb{R}$ の最適化では，導関数 F' の零点を求めることに帰着される．

このように場面に応じて関数 f は異なるが，零点探索はそれぞれの場面で重要な役割を果たしている．ここでは，計算機を用いた関数の零点探索を考える．はじめに，関数の零点探索のための反復法として知られる二分法を以下に与える．

二分法

Step 0: $f(a) < 0$, $f(b) \geq 0$ を満たす初期点 a, b を選び，終了条件 $\varepsilon > 0$ を決める．

Step 1: a と b の中間点 $c := (a + b)/2$ を求める．もし，c が終了条件 $|f(c)| \leq \varepsilon$ を満たしていれば，c を解として終了する．

Step 2: もし，$f(c) < 0$ であれば $a \leftarrow c$ とし，$f(c) \geq 0$ であれば $b \leftarrow c$ として，Step 1 へ戻る．

二分法は，関数 f が $\mathbb{R}$ 上で連続であれば必ず収束する．実際，Step 1 の初期条件を満たす a, b を選べば，関数 f の零点は a と b の間に存在するので，a と b の間隔を繰り返し 1/2 に縮小していけば，中間点 c は以下の図で示すように関数 f の零点へ近づいていくことがわかる．

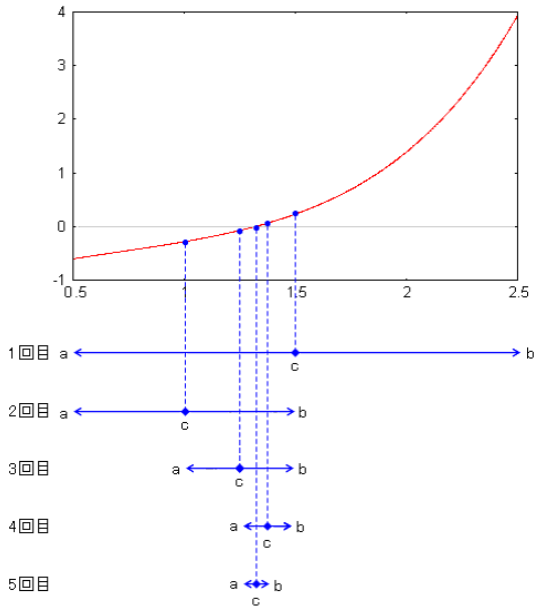

図 6.1: 二分法の生成点列

二分法は，アルゴリズムが簡易的であり，二つの異なる初期点 a, b の間に零点が存在すれば，必ず収束するという優れた性質を持つ．しかし，一般的には収束が遅いことで知られる．より収束が速い手法としては，次に与えるニュートン法が良く知られている．

ニュートン法

Step 0: 初期点 x_0 を選び，終了条件 $\varepsilon > 0$ を決める．$k := 0$ とする．

Step 1: ニュートン方程式

$$f'(x_k)\Delta x_k = -f(x_k)$$

を解き，Δx_k を求める．

Step 2: 点列を $x_{k+1} := x_k + \Delta x_k$ により更新する．もし，$|f(x_{k+1})| \leq \varepsilon$ を満たしていれば，x_{k+1} を解として終了する．

Step 3: $k \leftarrow k+1$ として，Step 1 へ戻る．

ニュートン法は，点 x_k の周りで関数 f を一次近似し，接線の方程式 $y = f'(x_k)(x-x_k)+f(x_k)$ を考える．この方程式と x 軸の交点を求め，更新点とする（つまり，$f'(x_k)(x-x_k)+f(x_k)=0$ を解き，$x = x_k - f(x_k)/f'(x_k)$ を更新点とする）ことで点列を生成する反復法である．一般的に収束は速いが，解の近くに初期点を選ばなければ収束しないということが知られている．

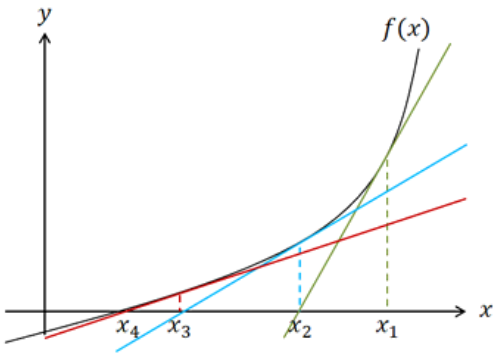

図 6.2: ニュートン法の生成点列

6.2.2 Python 導入

ここでは，Python の導入を行う．インターネット上のチュートリアルを参考にし，Anaconda3 を用いて Python をインストールせよ．例えば，

- `https://sukkiri.jp/technologies/ides/anaconda-win_install.html`

が参考となる．Python 3.0，数値計算ライブラリの Numpy，グラフ描写ライブラリの Matplotlib が PC にインストールされている場合は飛ばしても良い．

Python ヒント

Python はスクリプト言語なので，実行方法が 2 通りある：(a) 直接プログラムを 1 行ずつ打ち込む方法と，(b) プログラムファイルを作成し，インタプリタに一気に実行させる方法．(a) の場合は，簡単な計算の確認などのテストをするために便利であり，ターミナル上に `python` コマンドを入力することによってインタプリタが実行される．(b) の場合は，「`.py`」という拡張子のファイル（例えば，`test.py`）を用いて，ターミナル上に `python test.py` を入力し，プログラムを実行する．また，anaconda3 をインストールした際に，Spyder と呼ばれる Python 専用のエディターが同時にインストールされるので，こちらを用いるのも良い．

Python ヒント

本演習では，ライブラリ Numpy・Matplotlib が必要となる．インポートするために

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

を実行する，またはファイルに入力する．ここでは，Numpy の組込み関数を有効に使う．コンパイラ言語 C と違って，例えば，

```
w = np.random.rand(4)
```

を入力すると，ランダムな 4 次元ベクトルが `while` 文や `for` 文なしで生成される．同様に，

```
y = np.random.rand(4)
sum = w + y
```

を追加すると，2 つのランダムなベクトルの和が簡単に得られる．

課題 15 次の関数 $f\colon \mathbb{R} \to \mathbb{R}$ を考える．

$$f(x) := x^3 + 2x^2 - 5x - 6$$

Python を用いて，以下の課題に取り組め．

(a) 関数 f を $-10 \leq x \leq 10, -10 \leq y \leq 10$ の範囲で描写せよ．

(b) 二分法により関数 f の全ての零点を求めよ．ただし，二つの異なる初期点は，(a) で描写した関数 f の概形を参考にして適切に設定せよ．

(c) ニュートン法により関数 f の全ての零点を求めよ．ただし，(a) で描写した関数 f の概形を参考にして，零点の近くに初期点を設定せよ．

Python ヒント

グラフ描写は以下を参考にすると良い．例えば，$y = \sin(x)$ を $-5 \leq x \leq 5, -5 \leq y \leq 5$ の平面内に描写する際は，Numpy・Matplotlib をインポートしてから以下を入力する．

```
x = np.arange(-5,5,0.1)
y = np.sin(x)
plt.plot(x, y, 'red')
plt.xlim(-5,5)
plt.ylim(-5,5)
plt.axhline(0, c='black')
plt.axvline(0, c='black')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graph of y=sin(x)')
plt.show()
```

6.3 数理最適化の基礎と降下法

ここでは，まず数理最適化における基本を学ぶ．さらに，最も基本的な最適化手法として知られる降下法の概要を説明し，その具体例として最急降下法およびニュートン法を学ぶ．

6.3.1 関数の最大化・最小化

工学的分野に限らず，様々な分野において，与えられた関数の最小値，または最大値を求めることは重要である．例えば，企業においては，その利益を最大化することを目的としており，また，統計の分野における最小二乗法は，与えられたデータとの誤差を最小にする近似曲線を求めることを目的としている．

各分野において，その最小化または最大化しようとする関数は異なるが，ここでは，一般に $f\colon \mathbb{R}^n \to \mathbb{R}$ で与えられているとする．このとき，その関数の最小化問題は，

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in X \end{aligned} \tag{6.1}$$

として定式化される．これは,「集合 $X \subseteq \mathbb{R}^n$ の中から，関数値 $f(x)$ を最小とする x を求めよ」という問題である．ここで，f をこの最小化問題の**目的関数**と呼び，集合 X を**制約集合**と呼ぶ．さらに，制約条件を満たす点を**実行可能解**という．制約集合 X が $X = \mathbb{R}^n$ のとき，問題 (6.1) を**無制約最小化問題**という．本演習では，制約無し最小化問題のみを扱う．また，最大化問題については，最大化する関数が $\tilde{f}$ で与えられているとき，この問題は $f(x) := -\tilde{f}(x)$ の最小化問題へと同値に変形することができるため，ここでは最小化問題のみを考える．

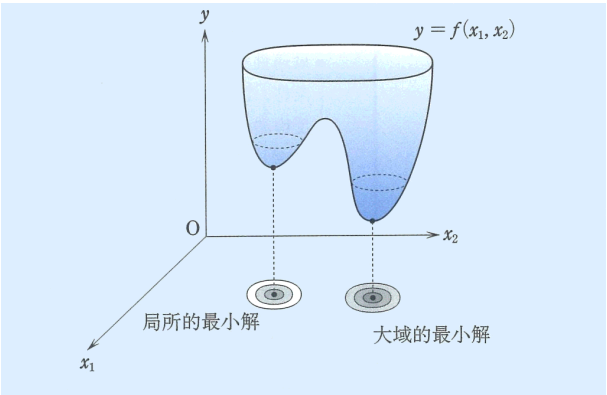

図 6.3: 大域的最小解，局所的最小解

最小化問題において，全ての実行可能解 x の関数値よりも大きくならない関数値をもつ点 $\bar{x} \in \mathbb{R}^n$ を，その問題の**大域的最小解**と呼ぶ (図 6.3)．つまり，次が成り立つ．

$$f(\bar{x}) \leq f(x) \quad \text{for all } x \in \mathbb{R}^n$$

しかし，大域的最小解であるかどうかの判定には，$\mathbb{R}^n$ 上の全ての点の情報が必要となるため，一般

的に大域的最小解を求めることは難しい．このため，次の局所的最小解という概念が重要となる．ここで，$\hat{x}$ が**局所的最小解**とは，$\hat{x}$ を含むある近傍 $N \subseteq \mathbb{R}^n$ を取ることができて，その近傍内では $f(\hat{x})$ よりも関数値が小さくなる実行可能解が存在しないことである (図 6.3)．つまり，$\hat{x}$ を含むある近傍 N が存在し，次を満たすことである．

$$f(\hat{x}) \leq f(x) \quad \text{for all } x \in N$$

もし，$N = \mathbb{R}^n$ ならば，局所的最小解は大域的最小解となる．関数 f が微分可能であるとき，

$$\nabla f(x) := \left(\frac{\partial f(x)}{\partial [x]_0}, \ldots, \frac{\partial f(x)}{\partial [x]_{n-1}} \right)^\top = 0$$

を満たす点 x のことを f の停留点と呼ぶ．ただし，$[x]_k$ $(k = 0, 1, \ldots, n-1)$ はベクトル x の第 k 成分であり，$\top$ は転置を表す．また，$\nabla f(x)$ は点 x における f の勾配ベクトルと呼ばれる．もし，$\hat{x}$ が局所的最小解であれば，$\hat{x}$ は f の停留点となる．しかし，f の停留点が f の局所的最小解であるとは限らない．局所的最大解や鞍点の場合もあるからである (図 6.4)．

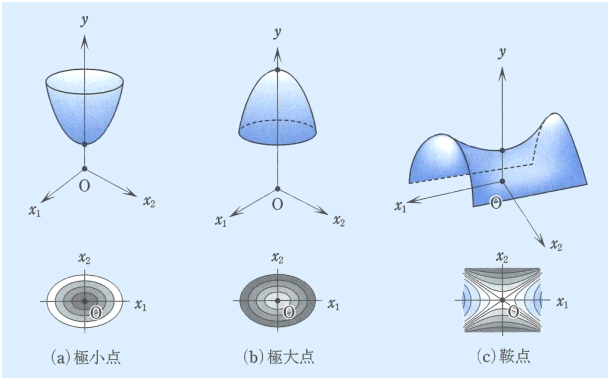

図 6.4: 局所的最大解と鞍点

本演習で扱う最急降下法およびニュートン法は，基本的に停留点を求める手法であり，必ずしも局所的最小値を求められるわけではない．しかしながら，大抵の問題に対しては，それらの手法によって局所的最小解を求めることができる．また，局所的最小値を求めることができる複雑な手法も，その根本的なところで，最急降下法やニュートン法を用いている．このため，それらの手法を理解することは重要である．

6.3.2 降下法

最急降下法やニュートン法は降下法と呼ばれる手法である．降下法は，関数の値を減少させる点列 $\{x^0, x^1, x^2, \ldots\}$ を生成していく反復法である[1]．つまり，

$$f(x^0) > f(x^1) > \cdots$$

となる点列 $\{x^k\}$ を生成する．この点列は

$$x^{k+1} := x^k + t_k d^k$$

で与えられる．ここで，d^k を**探索方向**，t_k を**ステップサイズ**と呼ぶ．降下法では，前提として，d^k は次の条件を満たすと仮定する．

$$\langle d^k, \nabla f(x^k) \rangle < 0$$

ここで，$\langle y, z \rangle$ はベクトル $y, z \in \mathbb{R}^n$ の内積を表している．つまり，$\langle y, z \rangle := \sum_{i=0}^{n-1} [y]_i [z]_i$ である．このような条件を満たしてるベクトル d^k のことを**降下方向**と呼ぶ．

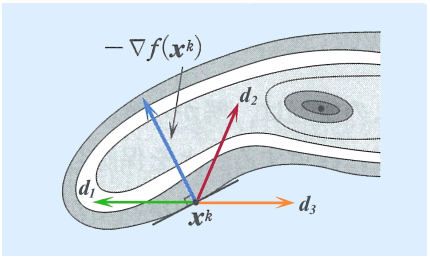

図 6.5: 降下方向

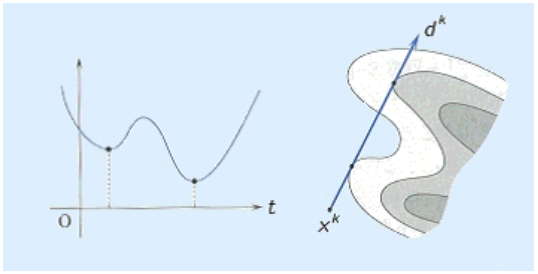

図 6.6: 直線探索

[1] $\mathbb{R}^n (n \geq 2)$ の点列を表す際は，上付き添え字を用いる．つまり，x^k は $\mathbb{R}^n$ の点列 $\{x^0, x^1, x^2, \ldots\}$ における k 番目の項を表す．なお，$\mathbb{R}$ や $\mathbb{R}^{n \times n}$ の点列を表す際は，冪乗との混同を避けるため下付き添え字を用いる．

図 6.5 は関数 f の等高線を表しており，図中の d_1 および d_2 は降下方向であるが，d_3 は降下方向ではない．降下方向へ適切に進めば，関数値は減少するが，進みすぎると増加する場合がある．そのため，$f(x^k) > f(x^{k+1})$ となるように，t_k をうまく設定する必要がある．この t_k を決める方法は**直線探索法**と呼ばれ，いくつか提案されているが，本テキストではバックトラック法を学ぶ．バックトラック法の詳細は 6.4 節で述べる．図 6.6（右）は探索方向を表す．図 6.6（左）は探索方向 d^k へ進んだ際の目的関数値の変化を表すグラフである．

バックトラック法を組み込んだ降下法を以下に与える．ただし，$\|z\| := (\sum_{i=0}^{n-1}[z]_i^2)^{1/2}$ である．

降下法

Step 0: 初期点 x^0，終了条件 $\varepsilon > 0$ を決める．$k := 0$ とする．

Step 1: もし，x^k が終了条件 $\|\nabla f(x^k)\| \leq \varepsilon$ を満たしていれば，x^k を解として終了する．

Step 2: 次式を満たす降下方向 d^k を定める．

$$\langle d^k, \nabla f(x^k)\rangle < 0$$

Step 3: バックトラック法を用いて，ステップサイズ t_k を決定する．

Step 4: 点 x^k を $x^{k+1} := x^k + t_k d^k$ と更新する．$k \leftarrow k+1$ として，Step 1 へ戻る．

注意 6.3.1 Step 1 における終了条件のパラメータ ε は，$\varepsilon = n \times 10^{-6}$ のように設定されることが多い．これは，一般に問題の次元 n が大きくなるにつれて，計算誤差が大きくなるため，この誤差を許容するという意味がある．

注意 6.3.2 反復回数 (k) が十分大きいとき，「解が求まらない」と判断する．よって，許容される最大反復回数の設定が必要である．

この降下法によって生成される点列 $\{x^k\}$ に対して，以下で与える収束に関する定理が示されている．証明に興味がある方は文献 [1] を読んでもらいたい．

定理 6.3.1 点列 $\{x^k\}$ および $\{d^k\}$ は，それぞれ降下法により生成される有界列とし，$\{d^k\}$ は降下方向の列であると仮定する．また，ある正の定数 γ が存在し，全ての $k \in \mathbb{N} \cup \{0\}$ に対して，以下の不等式が成り立つと仮定する．

$$\langle d^k, \nabla f(x^k)\rangle \leq -\gamma \|\nabla f(x^k)\|^2$$

このとき，点列 $\{x^k\}$ の任意の集積点 x^* は，関数 f の停留点となる．つまり，$\nabla f(x^*) = 0$ が成り立つ．

この定理により，上手に降下方向を選んでやれば，問題の停留点 (多くの場合，局所最小解) を求めることができる．次節以降で紹介する最急降下法とニュートン法は，この定理の降下方向に関する仮定を満たす手法である．

6.3.3 最急降下法とニュートン法

ここでは，最急降下法およびニュートン法について説明をする．

最急降下法は，点 x^k が与えられたときに，降下方向として以下を用いる手法である．

$$\text{[最急降下方向]} \qquad d^k := -\nabla f(x^k)$$

いま，$\nabla f(x^k) \neq 0$ と仮定する．(もし，$\nabla f(x^k) = 0$ であれば，点 x^k は既に f の停留点となっている．) このとき，

$$\langle d^k, \nabla f(x^k)\rangle = -\|\nabla f(x^k)\|^2 < 0$$

が成り立つため，d^k は降下方向であり，定理 6.3.1 の仮定を満たすことがわかる．(実際，上記の等号により，定理 6.3.1 における正の定数 γ として，$\gamma = 1$ の存在が確かめられる．)

一方，ニュートン法は，x^k が与えられたときに，降下方向として以下を用いる手法である．

$$\text{[ニュートン方向]} \qquad d^k := -\nabla^2 f(x^k)^{-1} \nabla f(x^k)$$

ただし，$\nabla^2 f(x)$ は x における f のヘッセ行列を表しており，

$$\begin{bmatrix} \frac{\partial^2 f(x)}{\partial [x]_0^2} & \frac{\partial^2 f(x)}{\partial [x]_0 [x]_1} & \cdots & \frac{\partial^2 f(x)}{\partial [x]_0 [x]_{n-1}} \\ \frac{\partial^2 f(x)}{\partial [x]_1 [x]_0} & \frac{\partial^2 f(x)}{\partial [x]_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial [x]_1 [x]_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial [x]_{n-1} [x]_0} & \frac{\partial^2 f(x)}{\partial [x]_{n-1} [x]_1} & \cdots & \frac{\partial^2 f(x)}{\partial [x]_{n-1}^2} \end{bmatrix}$$

で与えられる．ここで，$\{\nabla^2 f(x^k)\}$ は有界かつ一様正定値（詳細は 6.5 節で解説する）であると仮定する．これは，ある $\mu > 0$ が存在して，全ての $k \in \mathbb{N} \cup \{0\}$ と $v \in \mathbb{R}^n$ に対して，

$$\frac{1}{\mu}\|v\|^2 \leq \langle \nabla^2 f(x^k)^{-1} v, v \rangle \leq \mu \|v\|^2$$

が成り立つことと同値である．このとき，すべての k に対して，$\nabla f(x^k) \neq 0$ であれば，

$$\begin{aligned} \langle d^k, \nabla f(x^k) \rangle &= -\langle \nabla^2 f(x^k)^{-1} \nabla f(x^k), \nabla f(x^k) \rangle \\ &\leq -\frac{1}{\mu}\|\nabla f(x^k)\|^2 \\ &< 0 \end{aligned}$$

となるため，d^k は降下方向であり，定理 6.3.1 の仮定を満たすことがわかる．

課題 16 関数 $f\colon \mathbb{R} \to \mathbb{R}$ を以下で定義する．

$$f(x) := \frac{1}{3}x^3 - x^2 - 3x + \frac{5}{3}$$

Python を用いて，

(a) 最急降下法を実装し，関数 f の停留点を求めよ．ただし，初期点は $x_0 := 1/2$ と設定し，各反復 $k \in \mathbb{N} \cup \{0\}$ において，ステップサイズは $t_k := 1/(k+1)$ を採用せよ．

(b) ニュートン法を実装し，関数 f の停留点を求めよ．ただし，初期点は $x_0 := 5$ と設定し，各反復 $k \in \mathbb{N} \cup \{0\}$ において，ステップサイズは $t_k := 1$ を採用せよ．

6.4 直線探索と最適化手法の収束性

ここでは，前半で 6.3 節の降下法の概説で触れた直線探索法について学ぶ．特に，直線探索法の一種であるバックトラック法を詳説する．後半では，反復法の収束性について学ぶ．

6.4.1 直線探索法

直線探索法は，ステップサイズ $t_k > 0$ を決定するための手法であり，点 x_k を x_{k+1} へ更新する際に，$f(x_{k+1}) < f(x_k)$ を満たすようにするために行う．これまでにいくつかの直線探索法が提案されているが，以下に与えるバックトラック法は理論的にも実用的にも優れていることが知られている．

バックトラック法

Step 0: $\xi \in (0,1)$，$\rho \in (0,1)$，初期ステップサイズ $\bar{t} > 0$ を選ぶ．$t = \bar{t}$ とする．

Step 1: 次式を満たすまで $t \leftarrow \rho t$ とする．

$$f(x^k + td^k) \leq f(x^k) + \xi t \langle d^k, \nabla f(x^k) \rangle$$

Step 2: $t_k = t$ として終了．

Step 1 の不等式は**アルミホ条件**と呼ばれ，その意味は図 6.7 で表す．横軸はステップサイズ t，縦軸は $\varphi(t) := f(x^k + td^k)$ を表す．原点における φ の接線は $y = f(x^k) + t\langle d^k, \nabla f(x^k) \rangle$ である．アルミホ条件を満たすステップサイズを選ぶことは，φ の接線の傾きを緩和して得られる直線 $y = f(x^k) + \xi t \langle d^k, \nabla f(x^k) \rangle$ よりも関数値が小さくなる t の区間からステップサイズを選ぶことに対応していると言える．ξ を小さく設定すれば，アルミホ条件を満たすステップサイズを速く見つけることができる．

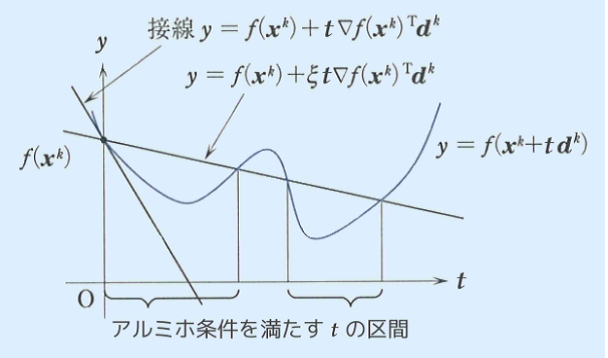

図 6.7: アルミホ条件を満たすステップサイズ

6.4.2 反復法の収束性について

ここまで，最急降下法およびニュートン法について学び，定理 6.3.1 によって，どちらの手法を用いても問題の解（多くの場合，局所最小解）を求められることがわかった．しかし，ここまでの議論では，どれだけ速く解を得ることができるかはわからない．これについて議論するためには，反復法の解への収束の速さに関する概念が必要となる．

ここで，x^* を求める問題の解とし，$\{x_k\}$ は x^* へ収束するとしよう．このとき，ある $\kappa \in (0,1)$ と $m_0 \in \mathbb{N}$ が存在して，

$$\|x^{k+1} - x^*\| \leq \kappa \|x^k - x^*\| \quad \text{for all } k \geq m_0$$

が成り立つとき，点列 $\{x^k\}$ は解 x^* に **1 次収束**するという．一方，

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} < \infty$$

が成り立つとき，点列 $\{x^k\}$ は解 x^* に **2 次収束**するという．これは，ある $\eta > 0$ と $n_0 \in \mathbb{N}$ が存在して，

$$\|x^{k+1} - x^*\| \leq \eta \|x^k - x^*\|^2 \quad \text{for all } k \geq n_0$$

となることを意味しており，一旦 $\|x^k - x^*\| < 1$ となると，非常に速く収束する．

ここで，次の点列 $\{y^k\}$ と $\{z^k\}$ を考えてみよう．

$$y_k = 0.5^k, \quad z_k = 0.5 z_{k-1}^2$$

ただし，$z_0 = 1$ とする．点列 $\{y_k\}$ は，0 に 1 次収束することが確かめられる．一方, 点列 $\{z_k\}$ は，0 に 2 次収束することが確かめられる．これらの点列を計算してみると，表 6.1 となる．表 6.1 より，2 次収束する点列は，1 次収束する点列に比べて，極めて速い収束性を持つことがわかる．

ここまでに学んだ最急降下法は，適切な仮定の下で一次収束することが示されている．しかし，最小化する関数の一階微分（つまり一次の情報）しか用いていないため，二次収束しないことが多い．一方，ニュートン法は関数のヘッセ行列（つまり二次の情報）を用いた方法であり，適切な仮定の下で二次収束することが知られている．よって，最急降下法とニュートン法の解への収束性を比較すると，一般にニュートン法に軍配が上がる．

表 6.1: 1 次収束と 2 次収束

k	y_k	z_k
0	1	1
1	0.5	0.5
2	0.25	0.125
3	0.125	0.0078125
4	0.0625	0.0000305
⋮	⋮	⋮

このように収束性の観点から両手法を見ると，ニュートン法の方が最急降下法に比べて有能な反復法であるかのように思える．しかし，ニュートン法は毎回の反復において，降下方向の生成のためにヘッセ行列 $\nabla^2 f(x^k)$ および勾配 $\nabla f(x^k)$ を計算し，ニュートン方程式 $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$ を解かなければならないため，勾配 $\nabla f(x^k)$ を計算するだけの最急降下法に比べて一反復当たりの計算負荷が大きい．従って，問題の規模が大きくなれば，最急降下法では計算が可能であっても，ニュートン法では計算が困難となることも有り得る．このように，解への収束が速いニュートン法の方が，最急降下法に比べて有能な反復法であるとは限らないことに注意して欲しい．

課題 17 次の 2 次元の最適化問題を考える．

$$\begin{aligned} \text{minimize} \quad & f(x) := x_0^2 + e^{x_0} + x_1^4 + x_1^2 \\ & \qquad\qquad -2x_0x_1 + 3 \\ \text{subject to} \quad & x := (x_0, x_1)^\top \in \mathbb{R}^2 \end{aligned}$$

Python を用いて，

(a) 点 x が与えられたとき，目的関数の値 $f(x)$ を出力する関数を作成せよ．

(b) 点 x が与えられたとき，勾配ベクトル $\nabla f(x)$ を出力する関数を作成せよ．

(c) 点 x が与えられたとき，ヘッセ行列 $\nabla^2 f(x)$ を出力する関数を作成せよ．

Python ヒント

- 課題 17 を実施する前に，$f(x) := x_0^2 + x_1$ の場合を考える．このとき，

$$\nabla f(x) = \begin{bmatrix} 2x_0 \\ 1 \end{bmatrix}, \nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

である．

- 以下のコードをファイルに入力し，実行してみる．コードも理解しておく[a]．

```
#!/usr/bin/env python
# -*- coding:  utf-8 -*-

import numpy as np

def evalf(x):
  # 目的関数の値 f(x) を計算する
  f = x[0]**2 + x[1]
  return f

def evalg(x):
  # f の勾配を計算する
  g = np.array([2*x[0], 1])
  return g

def evalh(x):
  # f のヘッセ行列を計算する
  H = np.array([[2, 0], [0, 0]])
  return H

def main():
  x = np.array([0.3, 5])
  f = evalf(x)
  g = evalg(x)
  H = evalh(x)
  print('f =', f, '\ng =', g, \
    '\nH = \n', H)
if __name__ == '__main__':
  main()
```

[a] Python 3 を使った例である．

課題 18 Python を用いて，

(a) バックトラック法を用いた最急降下法を実装することで課題 17 の最適化問題を解き，最適解，最適値，および反復回数を与えよ．ただし，初期点は $x^0 = (1,1)^\top$ と設定し，またバックトラック法における ξ，ρ，$\bar{t}$ は，それぞれ $\xi = 10^{-4}$，$\rho = 0.5$，$\bar{t} = 1$ と設定せよ．

(b) バックトラック法を用いたニュートン法を実装することで課題 17 の最適化問題を解き，最適解，最適値，および反復回数を与えよ．ただし，初期点は $x^0 = (1,1)^\top$ と設定し，またバックトラック法における ξ，ρ，$\bar{t}$ は，それぞれ $\xi = 10^{-4}$，$\rho = 0.5$，$\bar{t} = 1$ と設定せよ．

6.5 最適化手法の実用的な修正方法と総合演習

6.3 節で学習したニュートン法は，$\{\nabla^2 f(x^k)\}$ が一様正定値であることを仮定していたが，この仮定は非常に強い仮定であり，成り立たないこともしばしばある．ここでは，この仮定を満たさない場合にも適用可能となるように，ニュートン法を修正する方法について学ぶ．また，Python を用いて最急降下法とニュートン法を実装し，いくつかの最適化問題を解く総合演習を行う．

6.5.1 ニュートン方向の修正

まず，対称行列の正定値性と固有値の関係について学ぶ．対称行列 $M \in \mathbb{R}^{n\times n}$ が正定値であるとは，

$$\langle Mv, v\rangle > 0 \quad \text{for all } v \in \mathbb{R}^n \backslash \{0\}$$

を満たすことである．以降，対称行列 M が正定値であることを $M \succ O$ と表す．次の定理は，正定値性と固有値の関係を表すものであり，非常に重要な性質である．

定理 6.5.1 以下の二つの命題は同値である．

- 対称行列 $M \in \mathbb{R}^{n\times n}$ が正定値である．
- 対称行列 $M \in \mathbb{R}^{n\times n}$ の全ての固有値 $\lambda_j(M)$ $(j = 1, \ldots, n)$ は正である

また，$\mathbb{R}^{n\times n}$ の点列 $\{M_k\}$ が有界かつ一様正定値であるとは，ある $\kappa > 0$ が存在して，全ての $k \in \mathbb{N}\cup\{0\}$ に対して，

$$\frac{1}{\kappa}\|v\|^2 \le \langle M_k v, v\rangle \le \kappa\|v\|^2 \quad \text{for all } v \in \mathbb{R}^n$$

が成り立つことであり，これは，

$$\frac{1}{\eta}\|v\|^2 \le \langle M_k^{-1} v, v\rangle \le \eta\|v\|^2 \quad \text{for all } v \in \mathbb{R}^n$$

と同値である．ただし，$\eta := 1/\kappa$ である．

6.3 節で学んだニュートン法は，「$\{\nabla^2 f(x^k)\}$ が有界かつ一様正定値である」と仮定したが，一様正定値性については成り立たないことも多々あるため，このような場合に対応するために，次に述べるような修正を行う．各反復 k に対して，ヘッセ行列 $\nabla^2 f(x^k)$ に単位行列 I の τ 倍を加え，次の方程式を考える．

$$\big(\nabla^2 f(x^k) + \tau I\big) d^k = -\nabla f(x^k) \tag{6.2}$$

ただし，$\tau > 0$ は $\nabla^2 f(x^k) + \tau I \succ O$ となるようにとる．もし，$\nabla^2 f(x^k) \succ O$ ならば，$\tau = 0$ とすれば良いので，$\nabla^2 f(x^k) \not\succ O$ の場合を考えよう．いま，$\nabla^2 f(x^k) + \tau I$ の固有値は $\lambda_j(\nabla^2 f(x^k)) + \tau$ $(j = 1,\ldots,n)$ となることを考慮すると，例えば τ として，$\nabla^2 f(x^k)$ の最小固有値の絶対値にある定数（10^{-1} や 10^{-2} など）を加えた値を設定すれば，$\lambda_j(\nabla^2 f(x^k)) + \tau > 0$ $(j = 1,\ldots,n)$ となり，定理 6.5.1 より $\nabla^2 f(x^k) + \tau I \succ O$ となる．よって，この τ を用いて，式 (6.2) から探索方向 d^k を生成する．このようなニュートン方向の修正は実用上，良く行われるものであり有用である．

6.5.2 総合演習

ここでは，Python を用いて，いくつかの最適化問題を解く演習を行う．

課題 19 Python を用いて，最急降下法およびニュートン法を実装することで次の最適化問題を解き，最適解および各手法の反復回数を与えよ．ただし，初期点は $x^0 = [2, 0]^\top$ と設定し，またバックトラック法における ξ，ρ，$\bar{t}$ は，それぞれ $\xi = 10^{-4}$，$\rho = 0.5$，$\bar{t} = 1$ と設定せよ．

$$\begin{array}{ll} \text{minimize} & f(x) := \displaystyle\sum_{i=0}^{2} f_i(x)^2 \\ \text{subject to} & x \in \mathbb{R}^2 \end{array}$$

ただし，$f_i(x) := y_i - [x]_0(1 - [x]_1^{i+1})$ $(i = 0, 1, 2)$ と定義し，$y_0 = 1.5$, $y_1 = 2.25$, $y_2 = 2.625$ である．（参考：最適値は 0，最適解は $[3, 0.5]^\top$ である．）

ヒント

課題 19 の f の勾配とヘッセ行列は，それぞれ次のようになる．

$$\begin{aligned} \nabla f(x) &= 2\textstyle\sum_{i=0}^{2} f_i(x)\nabla f_i(x) \\ \nabla^2 f(x) &= 2\textstyle\sum_{i=0}^{2} \big(f_i(x)\nabla^2 f_i(x) \\ &\qquad + \nabla f_i(x)\nabla f_i(x)^\top\big) \end{aligned}$$

参考文献

[1] D.P. Bertsekas, “Nonlinear Programming”, 2nd edition, Athena Scientific (1999).

[2] 福島雅夫,「新版・数理計画入門」，朝倉書店 (2011).

[3] J. Nocedal and S.J. Wright, “Numerical Optimization”, 2nd edition, Springer Verlag (2006).

[山川雄也]