数値解析ノート

椛島 健太

2024 年 5 月 16 日

目次

第 1 章　本書について　9

第 2 章　記号　11

第 3 章　特殊関数　13
3.1　Legendre 関数　13

第 4 章　行列演算　15
4.1　Moore-Penrose の一般化逆行列　15
4.2　QR 分解　15
4.3　特異値分解　16
4.4　線形方程式の反復解法　16
4.4.1　単純な反復による解法　16
4.4.2　Krylov 部分空間法　18
4.4.3　Algebraic Multigrid 法　18
4.4.4　Ordering（行または列の順番の入れ替えによる高速化）　22
4.4.5　反復解法の比較　22

第 I 部　数値積分　25

第 5 章　導入　27

第 6 章　1 次元の数値積分　29
6.1　Gauss 積分公式　29
6.1.1　Gauss-Legendre 公式　29
6.1.2　Gauss-Kronrod 積分公式　29
6.2　二重指数関数型公式　30
6.2.1　有限区間上の積分　30
6.2.2　半無限区間上の積分　31
6.2.3　無限区間上の積分　32
6.3　適用例　32
6.3.1　有限区間上の積分　32

第 7 章　多次元の数値積分　37
7.1　三角形上の数値積分　37

第 8 章　適応積分 39

第 II 部　最適化 41

第 9 章　導入 43

第 10 章　基本的な定義と性質 45

第 11 章　1 次元制約なし最適化 47
11.1　黄金比探索 47

第 12 章　勾配を用いた制約なし最適化 49
12.1　直線探索 49
12.2　最急降下法 51
12.3　Newton 法 51
12.4　準 Newton 法 51
12.5　共役勾配法 52

第 13 章　Downhill Simplex 法（勾配を用いない局所的な制約なし最適化） 53

第 14 章　大域最適化 55
14.1　Dividing Rectangles (DIRECT) 法 55
14.2　Adaptive Diagonal Curves 法 57

第 15 章　数値実験 59
15.1　対象としたアルゴリズム 59
15.2　制約のない凸関数の最適化 59
15.3　制約のない大域的最適化 62

第 III 部　求根アルゴリズム 65

第 16 章　導入 67

第 17 章　Newton-Raphson 法 69
17.1　1 次元の方程式の場合 69
17.1.1　平方根の算出 69
17.1.2　べき根の算出 70
17.2　一般の次元の方程式の場合 72
17.3　減速 Newton 法 72
17.4　最適化との関係 73

第 IV 部　常微分方程式の数値解法 75

第 18 章　導入 77

第 19 章 Runge-Kutta 法 79
19.1 埋め込み型公式 . . . 80
19.1.1 ステップ幅の自動更新 . . . 80
19.1.2 PI 制御によるステップ幅の自動更新 . . . 81
19.1.3 ステップ幅の簡易的な自動更新 . . . 82
19.1.4 初回のステップ幅の自動決定 . . . 82
19.2 埋め込み型でない公式による誤差の評価 . . . 83
19.3 陰的 Runge-Kutta 法における方程式の解法 . . . 84
19.3.1 半陰的公式の場合 . . . 84
19.3.2 陰的公式の場合 . . . 84
19.3.3 陰的公式の別の定式化 . . . 85
19.3.4 Jacobian の計算が困難な場合 . . . 85
19.4 陽的公式の例 . . . 89
19.4.1 古典的 Runge-Kutta 法 . . . 89
19.4.2 RKF45 公式 . . . 89
19.4.3 Euler 法 . . . 89
19.5 半陰的公式の例 . . . 90
19.5.1 田中 Formula . . . 90
19.5.2 SDIRK . . . 90
19.5.3 ESDIRK . . . 90
19.6 陰的公式の例 . . . 91
19.6.1 Butcher-Kuntzmann 公式 . . . 91
19.6.2 陰的 Euler 法 . . . 92
19.7 Rosenbrock 法 . . . 92

第 20 章 平均ベクトル場法 95

第 21 章 シンプレクティック積分法 97
21.1 陰的 Runge-Kutta 法 . . . 97
21.2 可分ハミルトニアンに対する陽的な分離型 Runge-Kutta 法 . . . 97

第 22 章 数値実験 101
22.1 対象とした数値解法 . . . 101
22.2 振り子の運動 . . . 102
22.3 Kaps の問題（硬い系の例） . . . 106

第 V 部 高精度演算 109

第 23 章 加算による丸め誤差の低減 111
23.1 Kahan Summation . . . 111
23.2 敗者復活算法 . . . 111

第 24 章 倍精度浮動小数点数による多倍長浮動小数点数 113

24.1 記号 . . . 113
24.2 基本演算 . . . 113
24.3 倍精度浮動小数点数による四倍精度演算 . . . 115
24.3.1 四則演算 . . . 115

第 VI 部 微分 117

第 25 章 導入 119

第 26 章 自動微分 121
26.1 前進型自動微分 . . . 121
26.2 後退型自動微分 . . . 121

第 VII 部 正則化 125

第 27 章 導入 127

第 28 章 Tikhonov 正則化 129
28.1 係数行列による解 . . . 129
28.2 特異値分解による解法 . . . 129
28.3 係数行列が横長の場合 . . . 131

第 29 章 一般化 Tikhonov 正則化 133
29.1 一般化特異値分解 . . . 134
29.2 単純な L2 ノルムによる Tikhonov 正則化への変換 . . . 134
29.2.1 行列 L が行と同じ数のランクを持つ場合の計算方法 . . . 136
29.2.2 行列 L が正則な正方行列の場合の計算方法 . . . 137
29.2.3 行列 L が一般の行列の場合の計算方法 . . . 138

第 30 章 L-curve 139
30.1 Tikhonov 正則化の場合 . . . 139
30.2 一般の場合 . . . 140

第 31 章 Generalized Cross Validation 141
31.1 導出 . . . 141
31.2 Tikhonov 正則化の場合における計算方法 . . . 142
31.3 一般の場合における計算方法 . . . 143

第 VIII 部 補間 145

第 32 章 導入 147

第 33 章 カーネル法 149
33.1 Tikhonov 正則化による導出 . . . 149

33.2 Gaussian Process . . . 150
33.3 RBF 補間 . . . 152
33.3.1 Wendland の Compactly Supported RBF . . . 152
33.4 多項式などによる項の追加 . . . 156
33.5 再生核ヒルベルト空間 . . . 156
33.5.1 Thin plate spline . . . 160
33.5.2 Spherical spline . . . 160
33.6 固有値分解による数値解法 . . . 161
33.6.1 追加の項がない場合 . . . 161
33.6.2 追加の項がある場合 . . . 161
33.7 パラメータ推定 . . . 162
33.8 大域最適解への応用 . . . 163

第 IX 部 付録 165

付録 A Lagrangian と Hamiltonian 167

付録 B 二重振り子 169

Todo list 170

参考文献 173

索引 177

第 1 章

本書について

本書では，私が数値解析関係で調査したことをまとめる．基本的には，実装のためにアルゴリズムやその理論を確認した結果をまとめることを目的としており，C++ で実装したものは Git リポジトリ [1] にて公開している．

第 2 章

記号

本書において共通で使用する記号を以下に示す．ただし，一部の章は分野独特の表記を用いる場合があるため，章専用の記号の一覧を持つ場合がある．

- $\mathbb{R}$ 実数の集合
- $\mathbb{C}$ 複素数の集合
- $\mathrm{Im}(z)$ 複素数 z の虚部
- A^* 行列 A のエルミート転置（随伴行列），または作用素 A の随伴作用素
- A^{-*} 行列 A のエルミート転置の逆行列
- $A^\dagger$ 行列 A の Moore-Penrose の一般化逆行列
- O 零行列
- I 単位行列
- $\mathrm{diag}(a_1, a_2, \ldots, a_n)$ 対角要素に $a_1, a_2, \ldots, a_n$ を持つ対角行列
- $\boldsymbol{a}_i$ 行列 A の i 列目
- $[A\boldsymbol{b}]_i$ ベクトル $A\boldsymbol{b}$ の第 i 要素
- $\|\boldsymbol{x}\|$ 一般のノルム
- $\|\boldsymbol{x}\|_2$ ベクトル $\boldsymbol{x}$ の 2-ノルム
- ∇f 関数 $f : \mathbb{R}^n \to \mathbb{R}$ の勾配
- $\nabla^2 f$ 関数 $f : \mathbb{R}^n \to \mathbb{R}$ の Hessian
- $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ 関数 $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ の Jacobian
- $A \succ O$ 正方行列 A は正定値である
- $A \succeq O$ 正方行列 A は半正定値である
- $A \succeq B$ $A - B \succeq O$ となる
- $f'(x)$ 1 変数関数 $f : \mathbb{R} \to \mathbb{R}$ の微分係数
- $\dot{\boldsymbol{y}}$ 関数 $\boldsymbol{y}(t)$ の一階微分
- $\ddot{\boldsymbol{y}}$ 関数 $\boldsymbol{y}(t)$ の二階微分
- $\dddot{\boldsymbol{y}}$ 関数 $\boldsymbol{y}(t)$ の三階微分
- $\ddddot{\boldsymbol{y}}$ 関数 $\boldsymbol{y}(t)$ の四階微分
- $P_n(x)$ Legendre 関数（3.1 節）
- $n!$ n の階乗
- δ_{ij} Kronecker のデルタ

第 3 章

特殊関数

この章では，数値計算でしばしば利用される特殊関数について，定義や基本的な性質，計算方法などを示す．

3.1 Legendre 関数

Legendre 関数は，$n = 0, 1, \ldots$ について次の式で表される [2, Section 5.2]．

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x-1)^n \tag{3.1}$$

Legendre 関数は n 次多項式で，区間 $[-1, 1]$ において次のような直交性を持つ [2, Section 6.3]．

$$\int_{-1}^{1} P_n(x) P_m(x) = \frac{2}{2n+1} \delta_{nm} \tag{3.2}$$

さらに，n 次の Legendre 関数は $n-1$ 次以下の任意の多項式 $f(x)$ と次のように直交する．

$$\int_{-1}^{1} P_n(x) f(x) = 0 \tag{3.3}$$

計算には次のような公式を用いる [2, Section6.3]．

$$P_{n+1}(x) = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x) \tag{3.4}$$

$$P_0(x) = 1 \tag{3.5}$$

$$P_1(x) = x \tag{3.6}$$

$$(1-x^2) P_n'(x) = n(P_{n-1}(x) - x P_n(x)) \tag{3.7}$$

0 から 5 次までの Legendre 関数をプロットしたものを図 3.1 に示す．

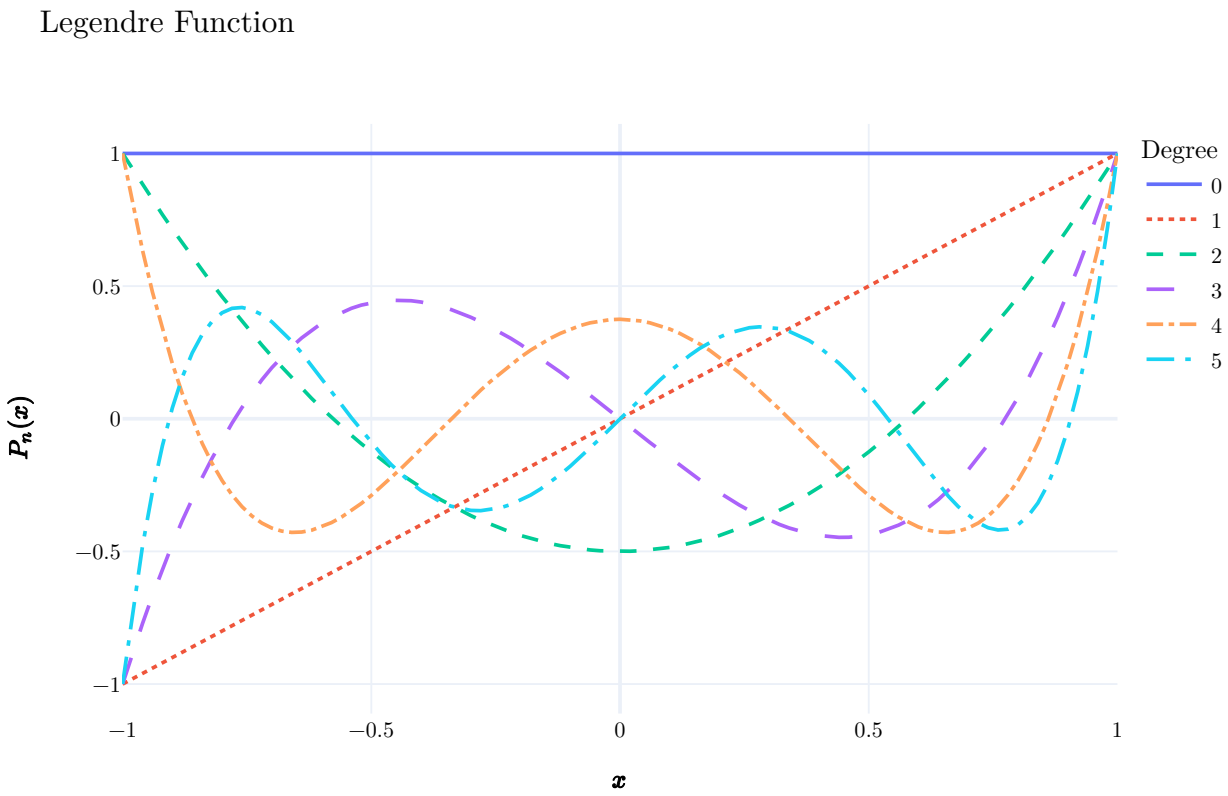


図 3.1: 0 から 5 次までの Legendre 関数

第 4 章

行列演算

本章では，基本的な行列演算についてまとめる．

4.1 Moore-Penrose の一般化逆行列

最小二乗法を理論的に扱う際に，Moore-Penrose の一般化逆行列がしばしば使用される．本資料でも使用するため，以下にその定義と性質を示す．

行列 $A \in \mathbb{C}^{m\times n}$ とベクトル $\boldsymbol{y} \in \mathbb{C}^m$ に対して，次の最適化問題を考える．

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{x}\|_2 \\ \text{s.t.} \quad & \|A\boldsymbol{x}-\boldsymbol{y}\|_2 = \min_{\boldsymbol{x}}\{\|A\boldsymbol{x}-\boldsymbol{y}\|_2\} \end{aligned}$$

この最適化問題の解は $\boldsymbol{x} = G\boldsymbol{y}$ のようにベクトル $\boldsymbol{y}$ に依らない行列 $G \in \mathbb{C}^{n\times m}$ を用いて表せる．この行列 G を Moore-Penrose の一般化逆行列と呼び，$A^\dagger$ で表す [3, 定義 3]．$\|A\boldsymbol{x}-\boldsymbol{y}\|_2$ を最小化するだけでは A の核空間が $\boldsymbol{0}$ 以外の要素を持つ場合に解が 1 つに定まらないが，$\|\boldsymbol{x}\|_2$ が最小となるものを選ぶことにより，解が 1 つに定まる．なお，定義から，$A^\dagger\boldsymbol{y}$ は A の核空間の成分を持たないことが示せる．

行列 A がランク n の場合，$\|A\boldsymbol{x}-\boldsymbol{y}\|_2$ を最小化する $\boldsymbol{x}$ は唯一であり，Moore-Penrose の一般化逆行列を用いて $\boldsymbol{x} = A^\dagger\boldsymbol{y}$ と表される．さらに，一般の行列 A においては，任意のベクトル $z \in \mathbb{C}^n$ を用いて

$$\boldsymbol{x} = A^\dagger\boldsymbol{y} + (I - A^\dagger A)\boldsymbol{z} \tag{4.1}$$

のように最小二乗解を表すことができる [3, 定理 2.3.1]．なお，$I - A^\dagger A$ は A の核空間への射影行列となっている．

Moore-Penrose の一般化逆行列は，ランクが m または n の場合に通常の逆行列を用いて次のように表現できる．

- $A \in \mathbb{C}^{m\times n}$ がランク m の場合

$$A^\dagger = A^*(AA^*)^{-1} \tag{4.2}$$

- $A \in \mathbb{C}^{m\times n}$ がランク n の場合

$$A^\dagger = (A^*A)^{-1}A^* \tag{4.3}$$

4.2 QR 分解

行列 $A \in \mathbb{C}^{m\times n}$ が $m \geq n$ かつランク n の場合に次のような分解を行うことができ，QR 分解と呼ばれる．

$$A = Q \begin{pmatrix} R \\ O \end{pmatrix} \tag{4.4}$$

ここで，$Q \in \mathbb{C}^{m\times m}$ はユニタリ行列であり，$R \in \mathbb{C}^{n\times n}$ は正則な上三角行列である．

行列 Q を $Q = (Q_1, Q_2)$ のように行列 $Q_1 \in \mathbb{C}^{m\times n}$ と行列 $Q_2 \in \mathbb{C}^{m\times(n-m)}$ に分割すると $A = Q_1 R$ となり，$A^\dagger = R^{-1}Q_1^*$ のように Moore-Penrose の一般化逆行列を求められる．

4.3　特異値分解

続いて，特異値分解についても触れておく．特異値分解には複数の定義があるが，ここでは行列 $A \in \mathbb{C}^{m\times n}$ の特異値分解は

$$A = U \begin{pmatrix} \Sigma & O \\ O & O \end{pmatrix} V^* \tag{4.5}$$

とする．ここで，$\Sigma \in \mathbb{R}^{r\times r}$ は正数の対角要素による対角行列で，$U \in \mathbb{C}^{m\times m}$ と $V \in \mathbb{C}^{n\times n}$ はユニタリ行列とする．

QR 分解と同様に $U = (U_1, U_2)$, $U_1 \in \mathbb{C}^{m\times r}$, $U_2 \in \mathbb{C}^{m\times(m-r)}$, $V = (V_2, V_2)$, $V_1 \in \mathbb{C}^{n\times r}$, $V_2 \in \mathbb{C}^{n\times(n-r)}$ のように行列を分割すると，$A = U_1 \Sigma V_1^*$ となり，$A^\dagger = V_1 \Sigma^{-1} U_1^*$ が示せる．

4.4　線形方程式の反復解法

ここでは，変数 $\boldsymbol{x} \in \mathbb{C}^n$ に関する線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ $(A \in \mathbb{C}^{m\times n}, \boldsymbol{b} \in \mathbb{C}^m)$ を反復的に解くアルゴリズムについて説明する．このようなアルゴリズムは以下のような場合に役に立つ．

- 行列のサイズ m, n が大きいが，行列 A が疎行列になっている場合，行列分解では大きな密行列ができてしまってコンピューターのメモリが足りなくなるが，疎行列 A を疎行列のまま扱える反復解法であれば使用できる．このような状況は，例えば偏微分方程式の数値計算において発生する．
- 行列 A 自身を計算するのは困難だが，行列 A を与えられたベクトル $\boldsymbol{x}$ にかけた $A\boldsymbol{x}$ は比較的容易に計算できる場合，$A\boldsymbol{x}$ さえ計算できれば使用できるタイプの反復解法を使用して線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解くことができる．このような状況は，例えば 19.3.4 節において扱う．

4.4.1　単純な反復による解法

まず，線形方程式について単純な反復による解法の例を示す．

Jacobi 法では，Algorithm 4.1 の反復を繰り返すことで解を求める．一方，Gauss-Seidel 法では，Algorithm 4.2 の反復を繰り返すことで解を求める．これらのアルゴリズムは，$A\boldsymbol{x} = \boldsymbol{b}$ を x_1 から順に 1 要素分ずつ解いていく．ただし，Jacobi 法では前の解のみから次の解の各要素を求めるのに対し，Gauss-Seidel 法では反復中の最新の解も使用するという違いがある．

さらに，Gauss-Seidel 法から派生したアルゴリズムとして，successive over-relaxation (SOR) 法（Algorithm 4.3），symmetric successive over-relaxation (SSOR) 法（Algorithm 4.4）がある．これらはパラメータ $\omega \in (0, 2)$ を持ち，SOR 法は $\omega = 1$ のとき Gauss-Seidel 法に一致する．SSOR 法は SOR 法で一回全ての変数を更新したあと，逆順に変数の更新を行う．

Algorithm 4.1 Jacobi 法の反復 [4]

procedure JacobiIterate(A, $\boldsymbol{b}$, $\boldsymbol{x}$)
  for $i = 1, 2, \ldots, n$ **do**
    $x'_i \leftarrow \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}$
  end for
  $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
end procedure

Algorithm 4.2 Gauss-Seidel 法の反復 [4]

procedure GaussSeidelIterate(A, $\boldsymbol{b}$, $\boldsymbol{x}$)
  for $i = 1, 2, \ldots, n$ **do**
    $x_i \leftarrow \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}$
  end for
end procedure

Algorithm 4.3 Successive over-relaxation (SOR) 法の反復 [4]

procedure GaussSeidelIterate(A, $\boldsymbol{b}$, $\boldsymbol{x}$)
  for $i = 1, 2, \ldots, n$ **do**
    $x'_i \leftarrow \omega \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} x'_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}} + (1 - \omega) x_i$
  end for
  $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
end procedure

Algorithm 4.4 Symmetric successive over-relaxation (SSOR) 法の反復 [4]

procedure GaussSeidelIterate(A, $\boldsymbol{b}$, $\boldsymbol{x}$)
  for $i = 1, 2, \ldots, n$ **do**
    $x'_i \leftarrow \omega \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} x'_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}} + (1 - \omega) x_i$
  end for
  for $i = n, n-1, \ldots, 1$ **do**
    $x_i \leftarrow \omega \dfrac{b_i - \sum_{j=1}^{i-1} A_{ij} x'_j - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}} + (1 - \omega) x'_i$
  end for
end procedure

Algorithm 4.5 BiCGstab [4]

procedure BiCGstab($A \in \mathbb{R}^{n\times n}, \boldsymbol{b} \in \mathbb{R}^n, \boldsymbol{x}_0 \in \mathbb{R}^n$)
  $\boldsymbol{r} \leftarrow \boldsymbol{b} - A\boldsymbol{x}_0$
  $\tilde{\boldsymbol{r}}_0$ を零ベクトルでない値に設定　▷ 残差 $\boldsymbol{r}$ にしておけば良い.
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
  $\boldsymbol{p} \leftarrow \boldsymbol{r}$
  $\rho \leftarrow \tilde{\boldsymbol{r}}_0^\top \boldsymbol{r}$
  loop
    $\boldsymbol{p}' \leftarrow A\boldsymbol{p}$
    $\mu \leftarrow \frac{\tilde{\boldsymbol{r}}_0^\top \boldsymbol{r}}{\tilde{\boldsymbol{r}}_0^\top \boldsymbol{p}'}$
    $\boldsymbol{s} \leftarrow \boldsymbol{r} - \mu\boldsymbol{p}'$
    $\boldsymbol{s}' \leftarrow A\boldsymbol{s}$
    $\omega \leftarrow \frac{\boldsymbol{s}^\top \boldsymbol{s}'}{\|\boldsymbol{s}'\|_2^2}$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \mu\boldsymbol{p} + \omega\boldsymbol{s}$
    $\boldsymbol{r} \leftarrow \boldsymbol{s} - \omega\boldsymbol{s}'$
    if $\|\boldsymbol{r}\|_2 < tolerance$ **then**
      return $\boldsymbol{x}$
    end if
    $\rho_{old} \leftarrow \rho$
    $\rho \leftarrow \tilde{\boldsymbol{r}}_0^\top \boldsymbol{r}$
    $\tau \leftarrow \frac{\rho\mu}{\rho_{old}\omega}$
    $\boldsymbol{p} \leftarrow \boldsymbol{r} + \tau(\boldsymbol{p} - \omega\boldsymbol{p}')$
  end loop
end procedure

4.4.2　Krylov 部分空間法

ここでは，Krylov 部分空間法と呼ばれる種類のアルゴリズムの例を示す.

対称行列の CG 法と PCG 法に触れておきたい.

行列 A が対称とは限らない正方行列である場合に使用できる反復解法の例として，BiCGstab (Algorithm 4.5) [*1]が挙げられる．ベクトル $\boldsymbol{x}$ について $A\boldsymbol{x}$ が計算できれば使用できるアルゴリズムとなっている.

4.4.3　Algebraic Multigrid 法

ここで，大規模で対称な疎行列 $A \in \mathbb{R}^{n\times n}$ を係数とする線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の解法の 1 つである Algebraic Multigrid (AMG) 法 [5] について説明する.

偏微分方程式の解法においては，対称となる領域（1 次元の弦，2 次元の水面や膜，3 次元の室内など）をグリッドに分け，各点ごとに近傍の点との位置関係などをもとにして係数行列 A が作られる．このような場合，方程式 $A\boldsymbol{x} = \boldsymbol{b}$ は近傍の点との関係のみを示すことになるが，細かいグリッドにおける方程式 $A\boldsymbol{x} = \boldsymbol{b}$ と荒いグリッドにおける方程式

[*1] 実装を意識して一部記法の変更を加えている.

Algorithm 4.6 Algebraic Multigrid (AMG) 法の準備（概要）[5, 6]

procedure AMGSetup(A)
  係数行列 A のグリッドの点のインデックスの集合を Ω_1 とする．
  $A_1 \leftarrow A$
  $m \leftarrow 1$
  loop
    A_m の値をもとに Ω_m から一段荒いグリッドに使用するインデックスを選択し，その集合を Ω_{m+1} とする．
    Ω_{m+1} 上のベクトルを Ω_m 上のベクトルに変換する補間の行列 P_{m+1}^m を生成する．
    Ω_m 上のベクトルを Ω_{m+1} 上のベクトルに変換する行列 $P_m^{m+1} = {P_{m+1}^m}^\top$ を用意する．
    Ω_{m+1} 上の係数行列 $A_{m+1} = P_m^{m+1} A_m P_{m+1}^m$ を算出する．
    if 係数行列 A_{m+1} が行列分解を適用できる程度に小さくなった場合 **then**
      return
    end if
    $m \leftarrow m + 1$
  end loop
end procedure

$A'\boldsymbol{x}' = \boldsymbol{b}'$ を用意しておくことで，領域全体の大まかな傾向は荒いグリッドで計算し，細かい部分は細かいグリッドで計算するといった使い分けができるようになり，細かいグリッドにおける詳細な解の計算を高速化することができる．そのようにして，複数の細かさの異なるグリッドを使い分けて最終的には細かいグリッドにおける詳細な解まで求められるようにしていくというのが基本的な Multigrid 法の考え方となっている．そのような細かさの異なるグリッドを実際のグリッドの形状から作るのではなく，細かいグリッドにおける係数行列の値をもとに荒いグリッドにおける係数行列を求めるようにして実現するのが AMG 法である．

AMG 法では，Algorithm 4.6 のようにして係数行列 A から反復的に荒いグリッドを生成していく．反復を停止する条件として，文献 [6] ではグリッドの点数が 1000 を下回ることが用いられている．1 段階荒いグリッドを求めるアルゴリズムは，Algorithm 4.7, Algorithm 4.8 からなる．

Algorithm 4.7 では，グリッドの点に対する評価値 λ_i が高い点を荒いグリッドの点に選択していく．その λ_i の基準となっている集合

$$S_i \leftarrow \left\{ j \in \Omega_m, j \neq i \;\middle|\; \left|[A_m]_{ij}\right| \geq \theta \max_{k \in \Omega_m, k \neq i} |[A_m]_{ik}| \right\} \tag{4.6}$$

における条件 $\left|[A_m]_{ij}\right| \geq \theta \max_{k \in \Omega_m, k \neq i} |[A_m]_{ik}|$ を満たすとき，i は j に強く接続している（strongly connected, strongly depends）という [5]．ここで，θ は $\theta \in (0, 1]$ を満たす定数で，実用的には 0.25 にしておけば良い [5]．アルゴリズム中に出てくる $S_i^\top$ は i に強く接続している点の集合となっており，強く接続している点が多いような点は優先的に荒いグリッドに含むようになっている．アルゴリズムの後半で λ_i の値が更新されているが，更新されたあとの値は $\lambda_i = \left|S_i^\top \cap U\right| + 2\left|S_i^\top \cap F\right|$ となっており，荒いグリッドに選択しないことにした点の集合 F に含まれる点との関係が強い点は次の荒いグリッドへ追加する点に選択されやすいようになっている．

Algorithm 4.8 では，各 $i \in F$ について，各 $j \in S_i$ は C にあるか，$C_i = C \cap S_i$ の少なくとも 1 点に強く接続しているという条件を満たすように荒いグリッドに選択する点を調整している．この条件を満たしていると効率よく計算できたという [5].

Algorithm 4.7 Algebraic Multigrid (AMG) 法における荒いグリッドの点の選択（ステップ 1）[5]

procedure AMGSelectCoarsePointsStep1(A_m, Ω_m)
  $C \leftarrow \emptyset$ ▷ 荒いグリッドの点の集合
  $F \leftarrow \emptyset$ ▷ 細かいグリッドのみの点の集合
  $U \leftarrow \Omega_m$ ▷ 未確認の点の集合
  for all $i \in \Omega_m$ **do**
    $S_i \leftarrow \left\{ j \in \Omega_m, j \neq i \;\middle|\; \left|[A_m]_{ij}\right| \geq \theta \max_{k \in \Omega_m, k \neq i} \left|[A_m]_{ik}\right| \right\}$ ▷ θ は $\theta \in (0, 1]$ な定数
  end for
  for all $i \in \Omega_m$ **do**
    $S_i^\top \leftarrow \left\{ j \in \Omega_m \;\middle|\; i \in S_j \right\}$
    $\lambda_i \leftarrow \left|S_i^\top\right|$
  end for
  while $U \neq \emptyset$ **do**
    U から λ_i が最も大きい i を選択する．
    $C \leftarrow C \cup \{i\}$, $U \leftarrow U \setminus \{i\}$
    for all $j \in S_i^\top \cap U$ **do**
      $F \leftarrow F \cup \{j\}$, $U \leftarrow U \setminus \{j\}$
      for all $k \in S_j \cap U$ **do**
        $\lambda_k \leftarrow \lambda_k + 1$
      end for
    end for
    for all $j \in S_i \cap U$ **do**
      $\lambda_j \leftarrow \lambda_j - 1$
    end for
  end while
end procedure

荒いグリッドを求めたあとは，補間の行列 P_{m+1}^m を求める．P_{m+1}^m の値として，文献 [6] では

$$[P_{m+1}^m]_{ij} = \begin{cases} 1 & i = j \in C \\ \frac{1}{|S_i^\top \cap C|} & i \in F, j \in S_i^\top \cap C \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

が挙げられている．

荒いグリッドを求めたあとは Algorithm 4.9 のような処理を反復的に行って解を更新する．アルゴリズム中の SMOOTH に使用する反復解法としては Gauss-Seidel 法（Algorithm 4.2）が挙げられる [5,6]．1 回の SMOOTH で行う Gauss-Seidel 法の反復の回数は 1 回で良い [6]．

AMG 法を既存の CG 法の前処理に使用した AMG-CG 法も考えられており，単純な前処理よりも効率よく方程式を解くことができるという [6]．

Algorithm 4.8 Algebraic Multigrid (AMG) 法における荒いグリッドの点の選択（ステップ 2）([5] をもとに一部変更)

procedure AMGSelectCoarsePointsStep2(A_m, C, F)
  $T = \emptyset$
  while $F \setminus T \neq \emptyset$ **do**
    $i \in F \setminus T$ を任意に選択する.
    $T \leftarrow T \cup \{i\}$
    $C_i \leftarrow S_i \cap C$ ▷ i に強く接続している荒いグリッド上の点
    $D_i^s \leftarrow S_i \setminus C_i$ ▷ i に強く接続している荒いグリッドにない点
    $\tilde{C}_i \leftarrow \emptyset$ ▷ 荒いグリッドへ追加する点の仮の置き場所
    for all $j \in D_i^s$ **do**
      if $S_j \cap C_i = \emptyset$ **then**
        if $\tilde{C}_i = \emptyset$ **then**
          $\tilde{C}_i \leftarrow \{j\}$
          $C_i \leftarrow C_i \cup \{j\}$, $D_i^s \leftarrow D_i^s \setminus \{j\}$
          for 文の処理をやり直す.
        else
          $C \leftarrow C \cup \{i\}$, $F \leftarrow F \setminus \{i\}$
          for 文を抜けて次の i の選択に移る.
        end if
      end if
    end for
    $C \leftarrow C \cup \tilde{C}_i$, $F \leftarrow F \setminus \tilde{C}_i$
  end while
end procedure

Algorithm 4.9 Algebraic Multigrid (AMG) 法による反復 [6]

procedure AMGIterate(m, $\boldsymbol{x}_m$, $\boldsymbol{b}_m$) ▷ m 番目に細かいグリッドで $A_m\boldsymbol{x}_m = \boldsymbol{b}_m$ を解くために反復する.
  if 最も細かいグリッドの場合 **then**
    行列分解を用いて $\boldsymbol{x}_m \leftarrow A_m^{-1}\boldsymbol{b}_m$ とする.
    return $\boldsymbol{x}_m$
  end if
  $\boldsymbol{x}_m \leftarrow \mathrm{SMOOTH}(A_m, \boldsymbol{x}_m, \boldsymbol{b}_m)$ ▷ 反復解法で $\boldsymbol{x}_m$ を更新する.
  $\boldsymbol{r}_m \leftarrow \boldsymbol{b}_m - A_m\boldsymbol{x}_m$
  $\boldsymbol{r}_{m+1} \leftarrow P_m^{m+1}\boldsymbol{r}_m$
  $\boldsymbol{e}_{m+1} \leftarrow \mathrm{AMGIterate(m} + 1, \boldsymbol{0}, \boldsymbol{r}_{m+1})$
  $\boldsymbol{e}_m \leftarrow P_{m+1}^m\boldsymbol{e}_{m+1}$ ▷ $A_m(\boldsymbol{x}_m + \boldsymbol{e}_m) = \boldsymbol{b}_m$ を荒いグリッドで解いたことになる.
  $\boldsymbol{x}_m \leftarrow \boldsymbol{x}_m + \boldsymbol{e}_m$
  $\boldsymbol{x}_m \leftarrow \mathrm{SMOOTH}(A_m, \boldsymbol{x}_m, \boldsymbol{b}_m)$
  return $\boldsymbol{x}_m$
end procedure

Algorithm 4.10 Cuthill-McKee Ordering [4, 7]

procedure CuthillMcKeeOrdering(A) ▷ $A \in \mathbb{C}^{n\times n}$ は疎行列
　$U \leftarrow \{1, 2, \ldots, n\}$ ▷ 未処理の点の集合
　点 $1, 2, \ldots, n$ について A の (i, j) 成分（$i \neq j$）A_{ij} が零でないときに点 i と点 j が接続されているようなグラフ G を考える．
　グラフ G における各点の次数 $d_i = \left|\left\{j \;\middle|\; A_{ij} \neq 0, i \neq j\right\}\right|$ を算出する．
　次数が最も小さい点 i について $S_0 \leftarrow \{i\}$, $U \leftarrow U \setminus \{i\}$ とする．
　$k \leftarrow 1$
　while $U \neq \emptyset$ **do**
　　S_{k-1} の点に接続されていて U に残っている点の集合を S_k とする．
　　$U \leftarrow U \setminus S_k$
　　$k \leftarrow k + 1$
　end while
　点 $1, 2, \ldots, n$ を $S_0, S_1, \ldots$ の順に並び替える．同じ集合 S_k に含まれる点同士は次数の小さい順に並び替える．
end procedure

4.4.4　Ordering（行または列の順番の入れ替えによる高速化）

疎行列の演算を高速化するために行または列の順番を入れ替えるアルゴリズムが考えられている．例えば，Cuthill-McKee Ordering (Algorithm 4.10) を用いて行と列を並び替えると行列の中央に非零要素が集まり，一部の疎行列解法に効果があることが知られている [4, 7]．また，Cuthill-McKee Ordering の結果を逆順に並び替えただけの Reverse Cuthill-McKee Ordering も考えられており，Cuthill-McKee Ordering より良い効果が得られる場合があるという [4, 7].

4.4.5　反復解法の比較

ここまでに紹介した反復解法の一部についてベンチマークを行った結果を示す．

Poisson 方程式 $-\triangle\varphi = f$ を 2 次元の格子上で離散化する．格子上の点は均等に間隔 d で並んでいるものとし，i 行目 j 列目（(i, j) と記す）の点の φ の値を $\varphi_{i,j}$ のように書くものとする．このとき，(i, j) における Laplacian は以下のように近似できる[*2]．

有限要素法の説明をして，導出過程を書くようにしたい．

$$-\triangle\varphi_{i,j} \approx \frac{8}{3d^2}\varphi_{i,j} - \frac{1}{3d^2}\left(\varphi_{i-1,j-1} + \varphi_{i-1,j} + \varphi_{i-1,j+1} + \varphi_{i,j-1} + \varphi_{i,j+1} + \varphi_{i+1,j-1} + \varphi_{i+1,j} + \varphi_{i+1,j+1}\right) \tag{4.8}$$

他のアルゴリズムも実装して比較する．

この近似をもとに領域 $[0, 1]^2$ 上の Poisson 方程式を解くための離散化された方程式 $A\boldsymbol{\varphi} = \boldsymbol{b}$（$\boldsymbol{\varphi}$ は $\varphi_{i,j}$ を並べたベクトル）を立て，以下のアルゴリズムで解いた．解くのにかかった時間を図 4.1 に示している．

- CG 法（ [8] の実装）

[*2] この近似は，文献 [7] を参考に格子上の各正方形内で双線形なテスト関数を用いた有限要素近似を行った結果である．

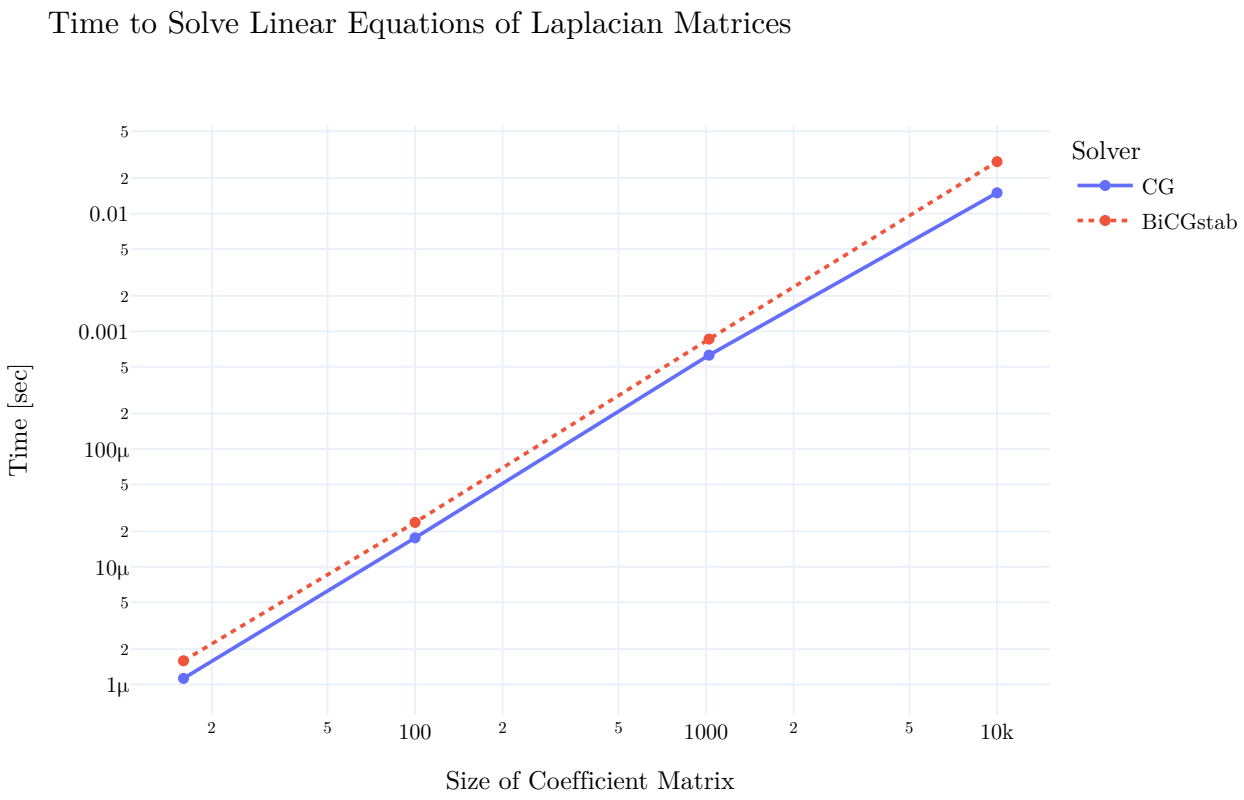


図 4.1: 2 次元格子上の Laplacian を離散化した行列を用いた反復解法のベンチマーク

- BiCGstab 法（ [8] の実装）

第 I 部

数値積分

第 5 章

導入

この部では，数値積分の手法をまとめる．

手計算で解析的に積分するのが困難な場合や，解析的な積分結果だけでは具体的な数値が不明な場合など，数値的に積分を行う必要がある場合は多々ある．何次元のどのような形状の領域でどのような関数を積分するかに応じて様々な積分手法が存在する．

第 6 章

1 次元の数値積分

本章では，1 次元における積分

$$\int_a^b f(x)dx \tag{6.1}$$

に対する数値積分の手法についてまとめる．

6.1 Gauss 積分公式

1 次元における積分に対する数値積分の手法の 1 つに Gauss 積分公式がある．（a, b は $-\infty$, ∞ でも良い．）
Gauss 積分公式では，次のように $x_1, x_2, \ldots, x_n$ 上での関数値の重み付き平均を用いる [9].

$$\int_a^b f(x)w(x)dx \approx \sum_{k=1}^{n} w_k f(x_k) \tag{6.2}$$

分点 $x_1, x_2, \ldots, x_n$ と重み $w_1, w_2, \ldots, w_n$ は，区間 (a, b) と重み関数 $w(x)$ に応じて決まる．

6.1.1 Gauss-Legendre 公式

a, b が有限な場合に利用できる Gauss-Legendre 公式では，Legendre 関数 $P_n(x)$ の n 個の零点を分点 x_k に用い，重み関数は次のように算出する [9].

$$w_k = \frac{2(1-x_k^2)}{(nP_{n-1}(x_k))^2} \tag{6.3}$$

分点 x_k の算出には Newton-Raphson 法（17 章）を用いることができる．初期値には

$$x_k \approx \cos\frac{\pi(k-0.25)}{n+0.5} \tag{6.4}$$

を使用すると良い [9]．また，$s_k = -x_{n-k}, w_k = w_{n-k}$ を用いて計算量を半減させることができる．

6.1.2 Gauss-Kronrod 積分公式

数値積分した際に，計算結果の誤差を評価したい場合がある[*1]．そこで，異なる近似値 2 つを用意し，その 2 つの差を誤差のオーダーの評価に用いる．同じ積分公式で次数を変えた 2 種類の分点 x_k と重み w_k を用いても良いが，計算

[*1] 利用例は適応積分（8 章）を参照.

効率を上げるため分点 x_k を使い回せるように考えられた「埋め込み型の公式」が存在する．埋め込み型の公式では，同じ分点で次のような 2 種類の近似値を求めることができる．

$$I = \sum_{k=1}^{n} w_k f(x_k) \tag{6.5}$$

$$I^* = \sum_{k=1}^{n} w_k^* f(x_k) \tag{6.6}$$

この 2 つの値の差を利用して誤差の評価を行う．

Gauss 積分公式において，追加の分点を加えることで追加の近似値を算出する手法として，Gauss-Kronrod 積分公式が考えられており，行列演算により追加の分点と重みを算出する方法も考えられている [10]．細かいアルゴリズムについては文献 [10] [*2]や Gauss-Legendre 公式に対する適応例 [1] を参照．

6.2　二重指数関数型公式

1 次元の数値積分において，Gauss 公式よりも困難な積分へ対応できる手法として，二重指数関数型公式（Double Exponential Formula, DE Formula）が存在する [9, 6.1 節 (b)], [12, Section 4.5]．二重指数関数型公式では，区間 (a, b) 上での積分において次の式による変数変換を行い，区間 $(-\infty, \infty)$ 上での積分にする．

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\tanh\left(\frac{\pi}{2}\sinh t\right) \tag{6.7}$$

tanh, sinh はどちらも指数関数で定義されるため，名前通り二重の指数関数による積分となっている．

二重指数関数型公式は，無限積分

$$\int_{-\infty}^{\infty} f(x)dx \tag{6.8}$$

が有限和

$$h\sum_{k=-\infty}^{\infty} f(kh) \tag{6.9}$$

でよく近似できるということに基づいている [9, 6.1 節 (b)]．二重指数関数型公式では，積分領域の端で被積分関数が発散する

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}}dx \tag{6.10}$$

のような積分でも，積分領域の端を無限遠へ移すことにより，Gauss-Legendre 積分公式よりも安定して数値積分を行うことができる．ただし，指数関数を何度も計算する必要があるため，二重指数関数型公式の方が計算時間は長くなる可能性がある．

6.2.1　有限区間上の積分

1 次元の有限区間上の積分

$$I = \int_a^b f(x)dx \tag{6.11}$$

[*2] 文献 [10] の前提として文献 [11] も参照しておくこと．特に，Gauss-Legendre 積分公式から埋め込み型の公式を算出するには文献 [11] の内容が必要．

を考える．変数変換

$$x = \varphi(t) \equiv \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\tanh\left(\frac{\pi}{2}\sinh t\right) \tag{6.12}$$

を適用する場合，

$$\frac{dx}{dt} = \varphi'(t) = \frac{\pi}{4}(b-a)\frac{\cosh t}{\cosh^2\left(\frac{\pi}{2}\sinh t\right)} \tag{6.13}$$

を用いて

$$I = \int_a^b f(x)dx = \int_{-\infty}^{\infty} f(\varphi(t))\varphi'(t) \tag{6.14}$$

のように近似でき，

$$T_h = h\sum_{k=-N}^{N} f(\varphi(kh))\varphi'(kh) \tag{6.15}$$

で近似できる．また，パラメータ h の最適値は

$$h \approx \frac{\log(2\pi N)}{N} \tag{6.16}$$

であり，この場合の数値積分の誤差のオーダーは

$$|T_h - I| \approx \exp(-kN/\log N) \tag{6.17}$$

となる [12, Section 4.5]．点数 N を 2 倍にすると有効桁数が約 2 倍になる．

なお，実装時はオーバーフローや桁落ちを防ぐため

$$\begin{aligned} b - x &= \frac{1}{2}(b-a)\left(1 - \tanh\left(\frac{\pi}{2}\sinh t\right)\right) \\ &= \frac{1}{2}(b-a)\left(1 - \frac{\exp\left(\frac{\pi}{2}\sinh t\right) - \exp\left(-\frac{\pi}{2}\sinh t\right)}{\exp\left(\frac{\pi}{2}\sinh t\right) + \exp\left(-\frac{\pi}{2}\sinh t\right)}\right) \\ &= \frac{1}{2}(b-a)\left(1 - \frac{1-\exp\left(-\pi\sinh t\right)}{1+\exp\left(-\pi\sinh t\right)}\right) \\ &= (b-a)\frac{\exp\left(-\pi\sinh t\right)}{1+\exp\left(-\pi\sinh t\right)} \end{aligned} \tag{6.18}$$

$$\begin{aligned} \varphi'(t) &= \frac{\pi}{4}(b-a)\frac{\cosh t}{\cosh^2\left(\frac{\pi}{2}\sinh t\right)} \\ &= \frac{\pi}{4}(b-a)\frac{4\cosh t}{\left(\exp\left(\frac{\pi}{2}\sinh t\right) + \exp\left(-\frac{\pi}{2}\sinh t\right)\right)^2} \\ &= \pi(b-a)\frac{\cosh t\exp(-\pi\sinh t)}{(1+\exp(-\pi\sinh t))^2} \end{aligned} \tag{6.19}$$

とすると良い [12, Section 4.5.2].

6.2.2 半無限区間上の積分

1 次元の半無限区間上の積分

$$I = \int_0^{\infty} f(x)dx \tag{6.20}$$

を考える．これに対する二重指数関数型公式の変数変換は

$$x = \varphi(t) \equiv \exp(\pi \sinh t) \tag{6.21}$$

である [12, Section 4.5.3]．微分すると

$$\frac{dx}{dt} = \varphi'(t) = \pi \exp(\pi \sinh t) \cosh t \tag{6.22}$$

となる．

半無限区間上の積分，および次節の無限区間上の積分においては，変数変換後の変数 t において区間 $[-4, 4]$ までの範囲で有限和による近似を行えば良い [12, 4.5.3]．

6.2.3　無限区間上の積分

1 次元の無限区間上の積分

$$I = \int_{-\infty}^{\infty} f(x) dx \tag{6.23}$$

を考える．これに対する二重指数関数型公式の変数変換は

$$x = \varphi(t) \equiv \sinh\left(\frac{\pi}{2} \sinh t\right) \tag{6.24}$$

である [12, Section 4.5.3]．微分すると

$$\frac{dx}{dt} = \varphi'(t) = \frac{\pi}{2} \cosh\left(\frac{\pi}{2} \sinh t\right) \cosh t \tag{6.25}$$

となる．

6.3　適用例

6.3.1　有限区間上の積分

以下の積分を数値積分で計算した．

$$\int_0^1 e^x dx = e - 1 \tag{6.26}$$

$$\int_{-1}^1 \sqrt{1 - x^2} dx = \frac{\pi}{2} \tag{6.27}$$

$$\int_{-1}^1 \frac{1}{\sqrt{1 - x^2}} dx = \pi \tag{6.28}$$

上から順に，

- 滑らかな関数の積分（最も簡単な例）
- 積分区間の端で急変する関数（滑らかな関数よりも難易度が高い）
- 積分区間の端で発散する関数（特異積分）

となっている．

まず，Gauss-Legendre 公式で積分したときの誤差を図 6.1 に示す．次数を増やすと精度が良くなる様子を確認できる．また，簡単な式 (6.26) の積分では誤差が小さいのに対し，積分区間の端で急変する式 (6.27) の積分では誤差が大きくなり，積分区間の端で発散する式 (6.28) の積分ではさらに誤差が大きくなっている．

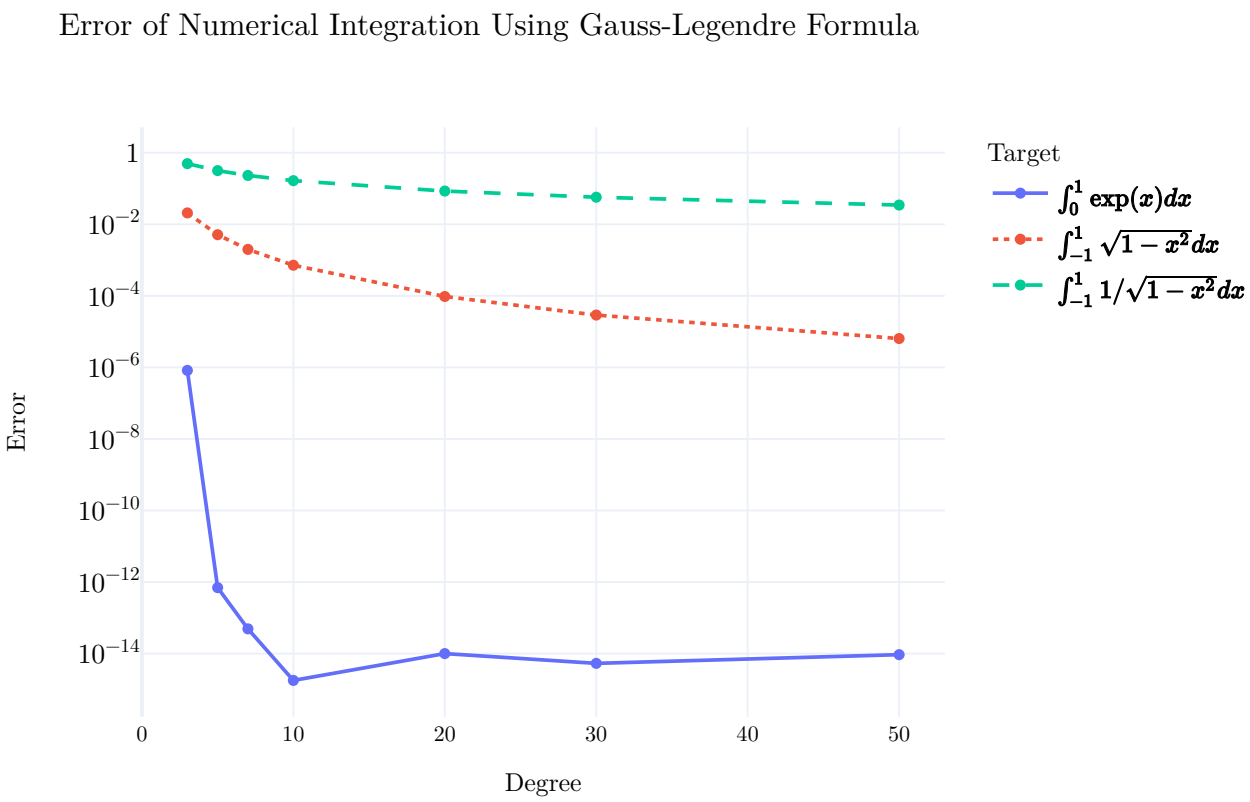


図 6.1: Legendre 関数の次数の異なる Gauss-Legendre 公式による数値積分の誤差

Gauss-Legendre-Kronrod 公式で適応積分したときの計算時間を図 6.2 に示す．計算結果の誤差はどれも 10^{-7} 程度となっており，その精度の値を計算するのにかかった時間が対象の積分や Legendre 関数の次数に応じて変化している．まず，対象の積分の難易度が高くなると計算時間が長くなった．積分区間を細かく分割して積分しないと十分な精度が出せないため，計算時間が長くなっていくものと考えられる．また，Legendre 関数の次数が小さい場合も計算時間が長くなっている．一回の積分公式の適用で得られる積分値の精度が低いため，積分区間を細かく分割して積分することになり，計算時間が長くなっていくものと考えられる．一方，次数を大きくしすぎた場合も計算時間が少し長くなっている．分割した個々の区間に対する積分の計算に時間がかかるためと考えられる．

二重指数関数型公式で積分したときの誤差を図 6.3 に示す．点数の多いときの結果がないのは，精度が倍精度浮動小数点数の桁数を超えてしまって誤差の値が 0 となってしまったためである．二重指数関数型公式は積分区間の端で被積分関数が急変したり発散したりするような難しい積分に強く，どの積分でも計算に用いる点数を増やした場合は誤差が同程度に小さくなる．

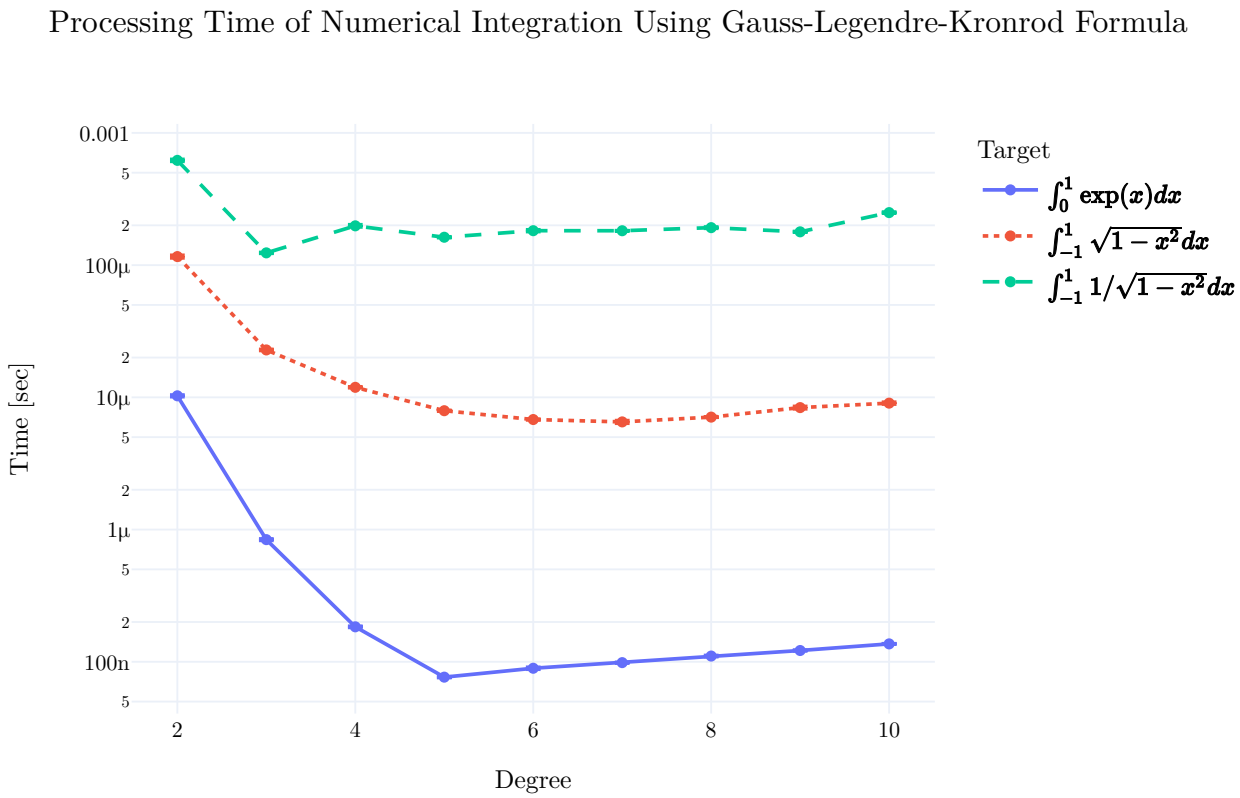


図 6.2: Legendre 関数の次数の異なる Gauss-Legendre-Kronrod 公式による数値積分の計算時間

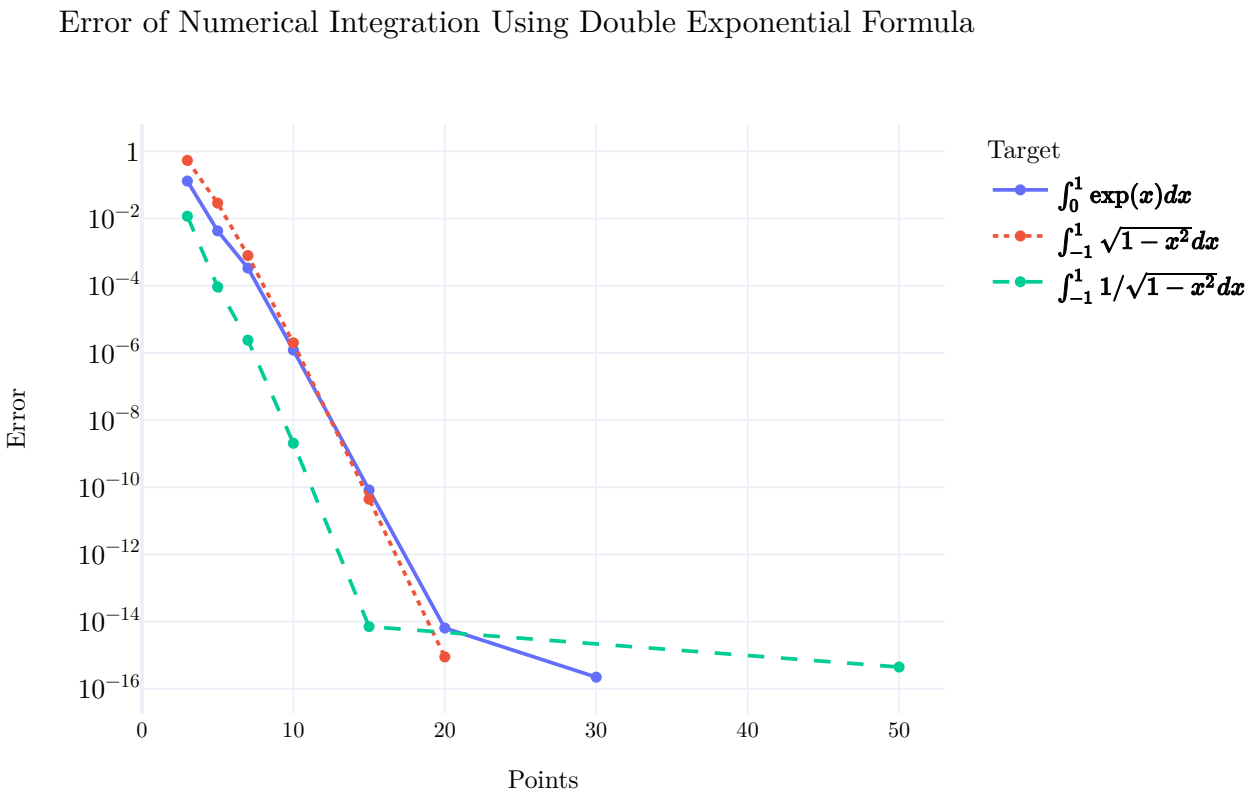


図 6.3: 点数の異なる二重指数関数型公式による数値積分の計算時間

第 7 章

多次元の数値積分

多次元領域上の積分は，変数変換で 1 次元ずつの多重積分にし，各次元に対して 1 次元の数値積分手法を適用すれば算出できる．しかし，よくある領域形状（三角形，直方体，四面体など）については，積分する領域の形状に合わせた様々な数値積分公式が研究されている [13].

7.1　三角形上の数値積分

2 次元平面上のメッシュや，3 次元空間内の曲面を表すメッシュなど，三角形が必要になる場面はしばしば存在する．そのような三角形での使用を目的とした数値積分手法の 1 つを紹介する．

三点 $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ からなる三角形[*1]において，三角形上の点 $\boldsymbol{p}$ を重心 $\boldsymbol{g}$ を用いて，

$$\boldsymbol{p} = \xi_1(\boldsymbol{a} - \boldsymbol{g}) + \xi_2(\boldsymbol{b} - \boldsymbol{g}) + \xi_3(\boldsymbol{c} - \boldsymbol{g}) + \boldsymbol{g} \tag{7.1}$$

のように $\xi_1 \geq 0, \xi_2 \geq 0, \xi_3 \geq 0, \xi_1 + \xi_2 + \xi_3 \leq 1$ なる ξ_1, ξ_2, ξ_3 の組で示すことができる．この原点を中心とした座標系は barycentric coordinates と呼ばれる．文献 [14] では，この barycentric coordinates を用いた対称的な積分公式 CUBTRI を算出している．また，CUBTRI は Gauss-Kronrod 積分公式（6.1.2 節）と同様に埋め込み型の公式であり，共通の分点を持つ複数の近似値を算出できる．

文献 [14] における積分公式は次のように計算する．

$$I_5[f] = \Delta \sum_{i=0}^{2} w_5^{(i)} I^{(i)}[f] \tag{7.2}$$

$$I_8[f] = \Delta \sum_{i=0}^{5} w_8^{(i)} I^{(i)}[f] \tag{7.3}$$

$$I^{(i)}[f] = \frac{1}{6} \sum_{\substack{k=1,2,3 \\ l=1,2,3 \\ k \neq l}} f(\xi_k^{(i)}, \xi_l^{(i)}) \tag{7.4}$$

ここで，$I_5[f]$, $I_8[f]$ はそれぞれ関数 f の 5, 8 次精度の積分となっている．Δ は三角形の面積であり，パラメータ $w_5^{(i)}$, $w_8^{(i)}$, $\xi_k^{(i)}$ は表 7.1 のようになっている．

[*1] 何次元空間の三角形かは問わない．

表 7.1: CUBTRI [14] のパラメータ（$\varphi = \sqrt{15}$, $\sigma = \sqrt{7}$）

i	$\xi_1^{(i)}$	$\xi_2^{(i)}$	$\xi_3^{(i)}$
0	$1/3$	$1/3$	$1/3$
1	$3/7 + 2\varphi/21$	$2/7 - \varphi/21$	$2/7 - \varphi/21$
2	$3/7 - 2\varphi/21$	$2/7 + \varphi/21$	$2/7 + \varphi/21$
3	$4/9 + \varphi/9 + \sigma/9 - \sigma\varphi/45$	$5/18 - \varphi/18 - \sigma/18 + \sigma\varphi/90$	$5/18 - \varphi/18 - \sigma/18 + \sigma\varphi/90$
4	$4/9 - \varphi/9 + \sigma/9 + \sigma\varphi/45$	$5/18 + \varphi/18 - \sigma/18 - \sigma\varphi/90$	$5/18 + \varphi/18 - \sigma/18 - \sigma\varphi/90$
5	$5/18 - \varphi/18 - \sigma/18 + \sigma\varphi/90$	$5/18 + \varphi/18 - \sigma/18 - \sigma\varphi/90$	$4/9 + \sigma/9$

i	$w_5^{(i)}$
0	$9/40$
1	$31/80 - \varphi/400$
2	$31/80 + \varphi/400$

i	$w_8^{(i)}$
0	$7137/62720 - 45\sigma/1568$
1	$-9301697/4695040 - 13517313\varphi/23475200 + 764885\sigma/939008 + 198763\varphi\sigma/939008$
2	$-9301697/4695040 + 13517313\varphi/23475200 + 764885\sigma/939008 - 198763\varphi\sigma/939008$
3	$102791225/59157504 + 23876225\varphi/59157504 - 34500875\sigma/59157504 - 9914825\varphi\sigma/59157504$
4	$102791225/59157504 - 23876225\varphi/59157504 - 34500875\sigma/59157504 + 9914825\varphi\sigma/59157504$
5	$11075/8064 - 125\sigma/288$

第 8 章

適応積分

積分に対して，積分公式を 1 回だけ適用して

$$\int_D f(\boldsymbol{x})dV \approx \sum_{i=1}^{N} w_k f(\boldsymbol{x}_k) \tag{8.1}$$

のように近似しただけでは十分な精度で計算できない場合がある．近似の精度が足りない場合に，精度の高い積分公式へ変更することで精度を上げることはできる．しかし，

$$\int_{-1}^{1} \sqrt{1-x^2}dx \tag{8.2}$$

のように積分領域の一部で大きな変動がある関数の積分をする場合は，変動が大きい箇所に対してより多くの分点 $\boldsymbol{x}_k$ を置いて詳細に計算を行った方がより良い近似値を得られると考えられる．そのような判断を自動で行って精度の高い近似値を得る手法として，適応積分が挙げられる．

適応積分では，近似誤差の推定値を見て必要に応じて

$$\int_0^1 f(x)dx = \int_0^{0.5} f(x)dx + \int_{0.5}^1 f(x)dx \tag{8.3}$$

のような領域の分割を行い，分割された個々の積分に対して積分公式を当てはめるという処理を繰り返すことにより，精度の出しにくい領域により多くの分点を配置した精度の良い近似値が得られるようにする [12, Section 4.7]．この処理は Algorithm 8.1 のように書ける．

なお，積分領域を分割していくことを前提としているため，積分公式自体の精度は高くなくても良い．ただし，積分公式の精度を下げすぎると，領域の分割を細かくする必要があり，時間がかかる可能性がある．積分公式の精度を選択できる場合は，全体の計算時間が短くなるちょうど良い精度を探しておくと良い．

適応積分における近似誤差の評価は，Gauss-Kronrod 積分公式（6.1.2）のような埋め込み型の公式の重要な適用例の 1 つである．

Algorithm 8.1 適応積分

```
1: procedure AdaptivelyIntegrate(f, D)          ▷ 関数 f を領域 D 上で適応積分する
2:     領域 D 全体で積分公式を適用する
3:     求められた近似値の誤差を評価する
4:     if 誤差が十分小さい，または領域が十分小さい then
5:         return 近似値
6:     else
7:         領域 D を分割する
8:         分割された個々の領域に対して AdaptivelyIntegrate を適用する
9:         各領域における積分の近似値を足し合わせる
10:        return 近似値の合計
11:    end if
12: end procedure
```

第 II 部

最適化

第 9 章

導入

この部では，最適化のアルゴリズムをまとめる．

最適化問題は一般に次のように書ける．

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) && (9.1)\\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} && (9.2)\\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} && (9.3) \end{aligned}$$

ここで，$\boldsymbol{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^r$ とする．また，ベクトル同士の「$\leq$」による比較は全ての要素において「$\leq$」の関係が成り立つことを意味する．

問題 (9.1) において，関数 f は 目的関数，$\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}$, $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$ は 制約条件と呼ばれ，制約条件を満たす中で目的関数を最小化する問題を示している．最小化した結果の変数 $\boldsymbol{x}$ は 最適解と呼ばれ，目的関数の値は 最適値と呼ばれる．

なお，最小化でなく最大化で定式化する場合もあるが，ここでは最小化に統一して説明する．

最適化のアルゴリズムは，対象とする問題に制約条件があるものとないものに分けられる．

第 10 章

基本的な定義と性質

定義 10.1 ([15, Section 6.4], [16, Section 3.1.1])**.** 関数 $f : \mathbb{R}^n \to \mathbb{R}$ が $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\forall \alpha \in [0,1]$ に対して

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}) \tag{10.1}$$

を満たす場合，関数 f を 凸関数（convex function）と呼ぶ.

定理 10.1 ([15, Section 6.4], [16, Section 3.1.3])**.** C^1 級関数 $f : \mathbb{R}^n \to \mathbb{R}$ が凸関数であるための必要十分条件は $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ に対して

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) \tag{10.2}$$

が成り立つことである.

定理 10.2 ([15, Section 6.4], [16, Section 3.1.3])**.** C^2 級関数 $f : \mathbb{R}^n \to \mathbb{R}$ が凸関数であるための必要十分条件は $\forall \boldsymbol{x} \in \mathbb{R}^n$ に対して

$$\nabla^2 f(\boldsymbol{x}) \succeq O \tag{10.3}$$

が成り立つことである.

定義 10.2 ([15, Section 6.4], [16, Section 3.1.1])**.** 関数 $f : \mathbb{R}^n \to \mathbb{R}$ が $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\forall \alpha \in [0,1]$ に対して

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) < \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}) \tag{10.4}$$

を満たす場合，関数 f は 狭義凸関数（strictly convex function）であるという.

定義 10.3 ([16, Section 9.1.2])**.** C^2 級関数 $f : \mathbb{R}^n \to \mathbb{R}$ がある定数 $m > 0$ を持ち，$\forall \boldsymbol{x} \in \mathbb{R}^n$ に対して

$$\nabla^2 f(\boldsymbol{x}) \succeq mI \tag{10.5}$$

を満たす場合，関数 f は 強凸関数（strongly convex function）であるという.

定理 10.3 ([16, Section 9.1.2])**.** C^2 級関数 $f : \mathbb{R}^n \to \mathbb{R}$ が強凸関数である場合，$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ に対して

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{m}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 \tag{10.6}$$

が成り立つ.

定理 10.4 ([16, Section 9.1.2])**.** C^2 級関数 $f : \mathbb{R}^n \to \mathbb{R}$ が強凸関数である場合，関数 f が最小となる $\boldsymbol{x} \in \mathbb{R}^n$ は一意的に定まる.

第 11 章

1 次元制約なし最適化

1 次元の変数 $x \in \mathbb{R}$ における関数 $f : \mathbb{R} \to \mathbb{R}$ の最小化においては，比較的簡単なアルゴリズムが存在する．

11.1　黄金比探索

黄金比探索 (golden section search, [12, 10.2]) では，区間を黄金比で分割していきながら最適解を探索する．
まず，区間 $[a, b]$ に対して中間点 c を

$$\frac{c-a}{b-a} = \omega \tag{11.1}$$

の比でとる（$0 < \omega < 1/2$）．そして，c と対称な位置に点 d をとる．つまり，

$$\frac{b-d}{b-a} = \omega \tag{11.2}$$

とする．このとき，

$$\frac{d-a}{b-a} = 1 - \frac{b-d}{b-a} = 1 - \omega \tag{11.3}$$

である．ここで，

- $f(c) < f(d)$ となった場合，最適解を探索する区間を $[a, b]$ から $[a, d]$ に更新する．
- $f(c) > f(d)$ となった場合，最適解を探索する区間を $[a, b]$ から $[c, b]$ に更新する．

のようにし，更新後の区間における中間点の相対位置が区間 $[a, b]$ における点 c と変わらないようにするため以下の方程式を立てる．

$$\frac{d-c}{d-a} = \omega, \qquad \frac{d-c}{b-c} = \omega \tag{11.4}$$

$b - a$ に対する比を用いることで，どちらも

$$\frac{1-2\omega}{1-\omega} = \omega \tag{11.5}$$

と変形でき，これを $0 < \omega < 1/2$ の制限のもとで解くと

$$\omega = \frac{3-\sqrt{5}}{2} \approx 0.3819660112501052 \tag{11.6}$$

となる．なお，このアルゴリズムの区間の分割において，以下のように黄金比が登場する．

$$\frac{b-a}{b-c} = \frac{1}{1-\omega} = \frac{1+\sqrt{5}}{2} \approx 1.618033988749895 \tag{11.7}$$

第 12 章

勾配を用いた制約なし最適化

ここでは，勾配を用いた最適化アルゴリズムをまとめる．

勾配を用いた最適化アルゴリズムでは，一般に Algorithm 12.1 のような手順で反復的に最適化を進めていく．更新方向の算出方法により，最急降下法，Newton 法，共役勾配法のような様々なアルゴリズムが存在する．

12.1 直線探索

まずは，Algorithm 12.1 におけるステップ幅 t_i を決定する直線探索の方法をまとめる．直線探索では $\nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i < 0$ となっている（つまり，$\boldsymbol{d}_i$ は目的関数が減少する方向になっている）ことを前提とする．

直線探索の方法として，以下のような方法が挙げられる．

- 厳密直線探索
- Backtracking line search [16, Section 9.2]
- 補間に基づく直線探索 [12, Section 9.7.1]

厳密直線探索は，更新後の目的関数の値 $f(\boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i)$ が最小となる t_i を探索する．

Backtracking Line Search [16, Section 9.2] は Armijo の条件 [15, Section 7.5]

$$f(\boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i) \leq f(\boldsymbol{x}_{i-1}) + \alpha t_i \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \tag{12.1}$$

を利用する．ここで，α は $\alpha \in (0, 1)$ を満たす定数であり，Armijo の条件は，図 12.1 のように十分小さい t_i を選択するための条件となっている．Backtracking Line Search では，$\alpha \in (0, 1/2)$ とし，Algorithm 12.2 のように t_i を初期値 1 から $\beta \in (0, 1)$ 倍していき，式 (12.1) を満たすものを探索する．一般に，パラメータ α, β は $\alpha \in [0.01, 0.3]$, $\beta \in [0.1, 0.8]$ の範囲で設定される [16, Section 9.2]．

文献 [12, Section 9.7.1] で示されている補間に基づく直線探索手法では，関数 $g_i(t_i) = f(\boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i)$ を補間して最小化する．Armijo の条件 (12.1) をある $t_i = \lambda_j$ に対して評価する際に算出される

$$g_i(0) = f(\boldsymbol{x}_{i-1}) \tag{12.2}$$

$$g_i'(0) = \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \tag{12.3}$$

$$g_i(\lambda_j) = f(\boldsymbol{x}_{i-1} + \lambda_j \boldsymbol{d}_i) \tag{12.4}$$

を用いて $g_i(t_i)$ を 2 次関数により補間する．

$$g_i(t_i) \approx g_i(0) + g_i'(0) t_i + \frac{g_i(\lambda_j) - g_i'(0)\lambda_j - g_i(0)}{\lambda_j^2} t_i^2 \tag{12.5}$$

Algorithm 12.1 勾配による最適化

procedure DescentMethod($f, \boldsymbol{x}_0$)
  for $i = 1, 2, \ldots$ **do**
    更新方向 $\boldsymbol{d}_i \in \mathbb{R}^n$ を算出する
    ステップ幅（更新方向に掛ける係数）t_i を決定する　▷ 通常 $f(\boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i) < f(\boldsymbol{x}_{i-1})$ とする
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i$
    if 終了条件を満たしている **then**
      return $\boldsymbol{x}_i$
    end if
  end for
end procedure

Algorithm 12.2 Backtracking Line Search [16, Section 9.2]

procedure BacktrackingLineSearch($f, \boldsymbol{x}_{i-1}, \boldsymbol{d}_i$)
  $t_i \leftarrow 1$
  while $f(\boldsymbol{x}_{i-1} + t_i \boldsymbol{d}_i) > f(\boldsymbol{x}_{i-1}) + \alpha t_i \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i$ **do**
    $t_i \leftarrow \beta t_i$
  end while
end procedure

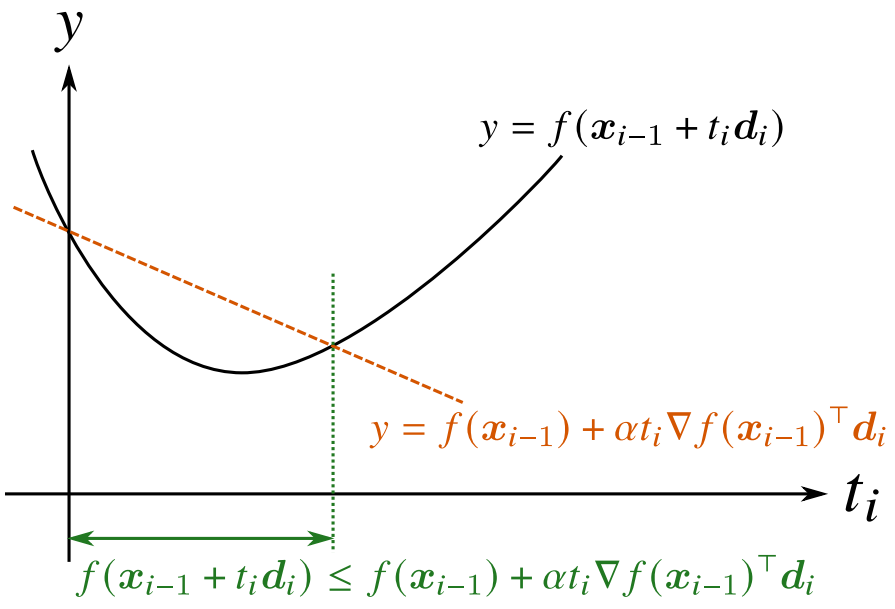


図 12.1: Armijo の条件（式 (12.1)）のイメージ

これを最小化することにより新しいステップ幅の候補

$$\lambda_{j+1} = -\frac{g_i'(0)\lambda_j^2}{2\left(g_i(\lambda_j) - g_i'(0)\lambda_j - g_i(0)\right)} \tag{12.6}$$

$$= -\frac{f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \lambda_j^2}{2\left(f(\boldsymbol{x}_{i-1} + \lambda_j \boldsymbol{d}_i) - \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \lambda_j - f(\boldsymbol{x}_{i-1})\right)} \tag{12.7}$$

が得られる．なお，$t_i = \lambda_j$ が Armijo の条件 (12.1) を満たしていない場合，

$$\begin{aligned} & f(\boldsymbol{x}_{i-1} + \lambda_j \boldsymbol{d}_i) - \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \lambda_j - f(\boldsymbol{x}_{i-1}) \\ >& f(\boldsymbol{x}_{i-1}) + \alpha \lambda_i \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i - \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \lambda_j - f(\boldsymbol{x}_{i-1}) \\ =& (\alpha - 1)\lambda_i \nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i \qquad (12.8) \\ >& 0 \qquad (12.9) \end{aligned}$$

となるため，ゼロ割りは発生せず，

$$0 < \lambda_{j+1} < \frac{\lambda_j}{2(1-\alpha)} \tag{12.10}$$

となる．さらに，2 回目以降の反復では，$g_i(\lambda_{j-1})$ の情報もも加えて 3 次関数による補間を行うことができる [12, Section 9.7.1] [*1]．

12.2 最急降下法

最急降下法では，更新方向を $\boldsymbol{d}_i = -\nabla f(\boldsymbol{x}_{i-1})$ とする．確実に目的関数の減少する方向を示しており，ここで示す他のアルゴリズムよりも更新方向の算出が簡単である．目的関数が強凸関数である場合において，最適解への収束が証明されている [16, Section 9.3.1]．

12.3 Newton 法

Newton 法では，目的関数が狭義凸関数である（つまり，Hessian $\nabla^2 f(\boldsymbol{x}_{i-1})$ が正定値である）場合を対象とし，更新方向を $\boldsymbol{d}_i = -\nabla^2 f(\boldsymbol{x}_{i-1})^{-1} \nabla f(\boldsymbol{x}_{i-1})$ とする．$\nabla^2 f(\boldsymbol{x}_{i-1})$ が正定値である場合，$\nabla^2 f(\boldsymbol{x}_{i-1})^{-1}$ も正定値になる[*2] ため，最適解でない $\boldsymbol{x}_{i-1}$ においては $\nabla f(\boldsymbol{x}_{i-1})^\top \boldsymbol{d}_i = -\nabla f(\boldsymbol{x}_{i-1})^\top \nabla^2 f(\boldsymbol{x}_{i-1})^{-1} \nabla f(\boldsymbol{x}_{i-1}) < 0$ となり，目的関数が減少する方向になっていることを確認できる．Newton 法の収束性については [16, Section 9.5.3, 9.6.4] にて議論されている．

12.4 準 Newton 法

Newton 法では，Hessian $\nabla^2 f(\boldsymbol{x}_{i-1})$ の逆行列が必要になるが，x^4 のように 2 階微分が 0 になる点があったり，Hessian の逆行列を安定的に計算できない点があったりする場合には使用できない．そこで，Newton 法の更新方向 $\boldsymbol{d}_i = -\nabla^2 f(\boldsymbol{x}_{i-1})^{-1} \nabla f(\boldsymbol{x}_{i-1})$ における Hessian を $\boldsymbol{d}_i = -H_{i-1} \nabla f(\boldsymbol{x}_{i-1})$ のように Hessian の代わりの行列で置き換える準 Newton 法と呼ばれるアルゴリズムがある．

準 Newton 法のうち，Davidon-Fletcher-Powell (DFP) 公式では次のように H_i を算出する [15, Section 9.3], [12, Section 10.9]．

$$H_{i+1} = H_i + \frac{\boldsymbol{p}_i \boldsymbol{p}_i^\top}{\boldsymbol{p}_i^\top \boldsymbol{q}_i} - \frac{H_i \boldsymbol{q}_i \boldsymbol{q}_i^\top H_i}{\boldsymbol{q}_i^\top H_i \boldsymbol{q}_i} \tag{12.11}$$

[*1] 2×2 行列の逆行列や 3 次関数の最小値の計算が必要になるため，容易ではない．

[*2] 対称行列 A が正定値である場合，A の固有値は正の実数である．A は固有値分解により $A = VDV^\top$（D は固有値による対角行列，V は直交行列）と書くことができるため，$A^{-1} = VD^{-1}V^\top$ となる．よって，A^{-1} の固有値も全て正の実数であり，A^{-1} は正定値である．

$$\boldsymbol{p}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \tag{12.12}$$
$$\boldsymbol{q}_i = \nabla f(\boldsymbol{x}_{i+1}) - \nabla f(\boldsymbol{x}_i) \tag{12.13}$$

初期値 H_0 を対称な正定値の行列にしておけば，全ての H_i が帰納的に正定値になる [15, Section 9.3]．H_i が正定値であれば，更新方向 $\boldsymbol{d}_i$ は目的関数の減少する方向になる．

また，同様の性質を持つ公式の 1 つとして，Broyden-Fletcher-Goldfarb-Shanno (BFGS) 公式が存在する [15, Section 9.4]．

$$H_i = B_i^{-1} \tag{12.14}$$
$$B_{i+1} = B_i + \frac{\boldsymbol{q}_i \boldsymbol{q}_i^\top}{\boldsymbol{q}_i^\top \boldsymbol{p}_i} - \frac{B_i \boldsymbol{p}_i \boldsymbol{p}_i^\top B_i}{\boldsymbol{p}_i^\top B_i \boldsymbol{p}_i} \tag{12.15}$$

逆行列を計算することで次のようにも書くことができる [12, Section 10.9]．

$$H_{i+1} = H_i + \frac{\boldsymbol{p}_i \boldsymbol{p}_i^\top}{\boldsymbol{p}_i^\top \boldsymbol{q}_i} - \frac{H_i \boldsymbol{q}_i \boldsymbol{q}_i^\top H_i}{\boldsymbol{q}_i^\top H_i \boldsymbol{q}_i} + \boldsymbol{q}_i^\top H_i \boldsymbol{q}_i \boldsymbol{v}_i \boldsymbol{v}_i^\top \tag{12.16}$$
$$\boldsymbol{v}_i = \frac{\boldsymbol{p}_i}{\boldsymbol{p}_i^\top \boldsymbol{q}_i} - \frac{H_i \boldsymbol{q}_i}{\boldsymbol{q}_i^\top H_i \boldsymbol{q}_i} \tag{12.17}$$

12.5　共役勾配法

共役勾配法では，

$$\boldsymbol{d}_1 = -\nabla f(\boldsymbol{x}_{i-1}) \tag{12.18}$$
$$\boldsymbol{d}_i = -\nabla f(\boldsymbol{x}_{i-1}) + \gamma_i \boldsymbol{d}_{i-1} \tag{12.19}$$
$$\gamma_i = \frac{(\nabla f(\boldsymbol{x}_{i-1}) - \nabla f(\boldsymbol{x}_{i-2}))^\top \nabla f(\boldsymbol{x}_{i-1})}{\|\nabla f(\boldsymbol{x}_{i-2})\|_2^2} \tag{12.20}$$

のように更新方向を算出する [15, Section 8.6]．γ_i については複数の形式があるが，ここで示している Polak-Ribiere 法は一般により良い結果が得られるという [15, Section 8.6], [12, Section 10.8]．Newton 法では計算量の多い逆行列の計算が必要だが，共役勾配法では計算量が変数の次元のオーダーに収まるため，各反復の計算時間を抑えられる．

第 13 章

Downhill Simplex 法（勾配を用いない局所的な制約なし最適化）

Downhill simplex 法 [12] では，制約なし最適化を対象とし，Algorithm 13.1 のように変数の空間における単体 (simplex) をルールに沿って反復的に動かしていくことで関数の最小値を求める手法である．関数値の大小のみで決まるため，関数の微分がなくとも最適化ができる．

Algorithm 13.1 Downhill simplex 法

```
1: procedure DownhillSimplex(f, x_0)
2:     変数の次元を n とし，n 次元の単体を構成する n+1 個の点 x_0, ..., x_n を決定する.
3:     loop
4:         単体における最大値以外の点からなる超面に対して対称な位置に最大値の点を移動する        ▷ Reflection
5:         if 移動した点の関数値が他の全ての点より小さい場合 then
6:             最大値の点を超面から離す                                  ▷ Reflection and expansion
7:         else if 移動した点がまだ最大値のままである場合 then
8:             最大値の点を超面に近づける                                ▷ Contraction
9:             if 移動した点がまだ最大値のままである場合 then
10:                最小値以外の全ての点を最小値に近づける                ▷ Multiple contraction
11:            end if
12:        else
13:            Reflection のみでこの反復における移動を完了する
14:        end if
15:        if 停止条件を満たしている場合 then
16:            return
17:        end if
18:    end loop
19: end procedure
```

第 14 章

大域最適化

勾配を用いた最適化アルゴリズムでは，局所的に目的関数が周りより小さい点（局所最適解）を見つけられるが，目的関数が最も小さくなる点（大域最適解）を見つけられるとは限らない．そこで，大域最適解を求める大域最適化のアルゴリズムが考えられている．本章では大域最適化のアルゴリズムについてまとめる．

14.1 Dividing Rectangles (DIRECT) 法

Dividing Rectangles (DIRECT) 法 [17] では，黄金比探索（第 11.1 節）と同様に領域を分割していきながら最適解を求める．ただし，黄金比探索と異なり大域最適解を求めるため，目的関数の値が小さくない点も一定の条件でさらに分割していく．

DIRECT 法では，探索領域を n 次元単位超立方体 $[0,1]^n$ とし，その上で定義される目的関数 $f : [0,1]^n \to \mathbb{R}$ を考える．目的関数は Lipschitz 連続であるとする．つまり，ある定数 K が存在し，任意の点 $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^n$ について

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq K\|\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{14.1}$$

が成り立つとする．アルゴリズムの進行とともに探索領域は超短形[*1]に分割されていくが，次に分割する超短形を決定するために，次のような「最適な可能性のある超短形」を定義する．

定義 14.1 ([17, Definition 4.1.])**.** 反復により探索領域が m 個の超短形に分割されているものとし，i 番目の超短形の中心点を $\boldsymbol{c}_i$ とする．j 番目の超短形が次の条件を満たすような定数 $\tilde{K}$ が存在する場合，最適な可能性のある超短形と呼ぶ．

$$f(\boldsymbol{c}_j) - \tilde{K}d_j \leq f(\boldsymbol{c}_i) - \tilde{K}d_i \qquad \text{for all } i = 1, \ldots, m \tag{14.2}$$

$$f(\boldsymbol{c}_j) - \tilde{K}d_j \leq f_{\min} - \varepsilon|f_{\min}| \tag{14.3}$$

ここで，$f_{\min} = \min\{f_i \mid i = 1, \ldots, m\}$ であり，d_i は i 番目の超短形の中心から端点までの距離であり，ε は正の実数の定数である．

$f(\boldsymbol{c}_j) - \tilde{K}d_j$ は Lipschitz 定数が $\tilde{K}$ である場合に j 番目の超短形の中で取り得る最小値を示す．そのため，式 (14.2) は，Lipschitz 定数が $\tilde{K}$ である場合に最も小さい値を取り得る超短形が j 番目の超短形であることを示す．式 (14.3) は，Lipschitz 定数が $\tilde{K}$ である場合に，j 番目の超短形で現状の大域最適解を少なくとも $\varepsilon|f_{\min}|$ だけ更新する可能性があることを示す．

超短形を分割する場合，各軸の区間の長さを比べて長い方向から分割することで，分割数の同じ超短形は同じ形状になるようにする．これにより，分割数の同じ超短形は定義 14.1 における d_i が同じになり，分割数の同じ超短形同士で

[*1] ここで扱う超短形 (hyper-rectangle) は $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$ のように区間の積で示される領域である．

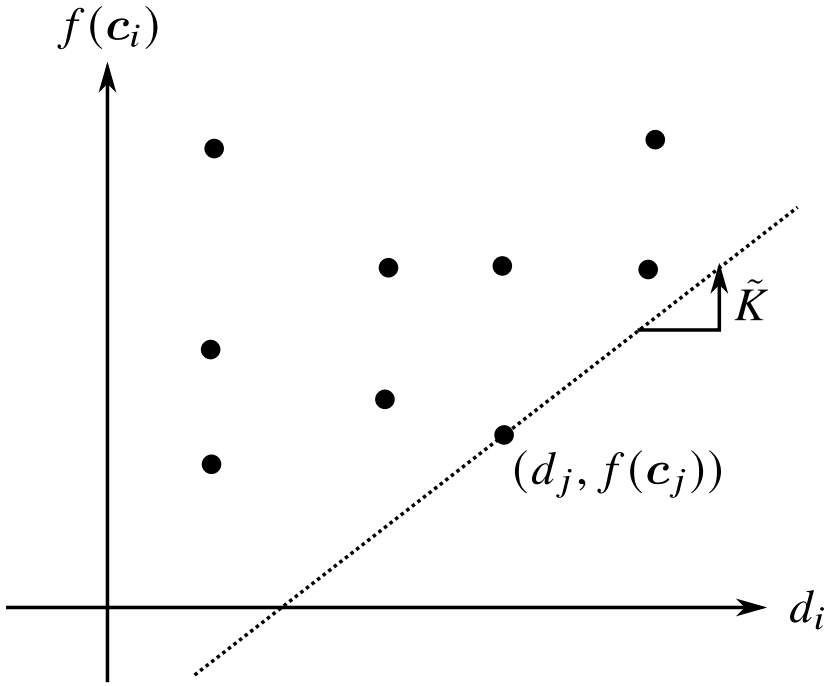


図 14.1: 式 (14.4) のイメージ

Algorithm 14.1 DIRECT 法

```
procedure DIRECT(f, , ε)
    単位超立方体 [0,1]^n を最初の超短形とする
    loop
        凸包探索を用いて式 (14.4) を満たす超短形を探索する
        式 (14.3) を満たさない超短形は除外する
        残った超短形を最も長い軸に沿って 3 つに分割する
        if 停止条件を満たす場合 then
            return
        end if
    end loop
end procedure
```

は式 (14.2) が単に目的関数の大小により評価できるようになる．また，式 (14.2) は

$$f(\boldsymbol{c}_i) \geq f(\boldsymbol{c}_j) + \tilde{K}(d_i - d_j) \tag{14.4}$$

のように書くこともでき，図 14.1 のように他の点が全て上に来るような直線を引けることを示す．このような点を探すアルゴリズムは凸包探索として知られる．

ここまでの議論により，DIRECT 法は Algorithm 14.1 のようになる．停止条件としては，反復回数が用いられる．DIRECT 法の派生としては以下のようなものが挙げられる．

- 各局所最適解への収束に重点をおいた DIRECT-l [18]

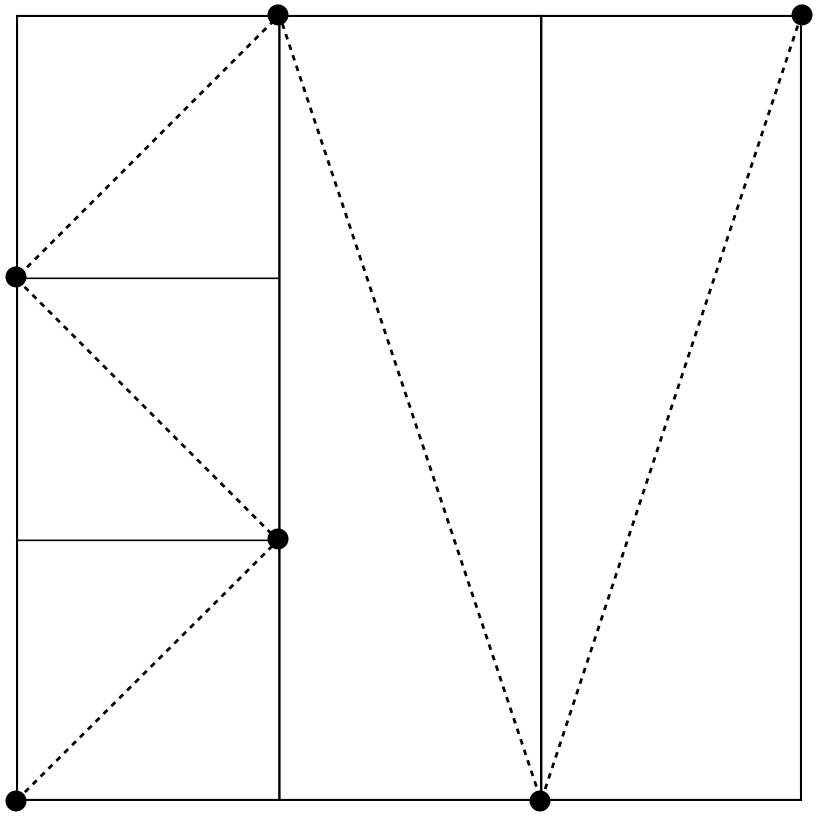


図 14.2: 文献 [21] で提案された領域分割と目的関数の値をとる頂点の例

- 対称な関数に最適化された SymDIRECT [19]

14.2 Adaptive Diagonal Curves 法

Adaptive Diagonal Curves 法 [20] は DIRECT 法（第 14.1 節）と同様に単位超立方体の探索領域を超短形に分割していくアルゴリズムである．しかし，分割する超短形を決定するために必要な目的関数の値は，超短形の中心でなく対角にある頂点でとる．目的関数の値をとる頂点が隣り合う超短形の間で共有できるように分割を行っていく手法 [21] を用いており，DIRECT 法よりも目的関数の値をとる点の数を減らすことができる．特に，困難な大域的最適化の問題において特に良い性能が出せるという [20, 22].

文献 [21] では，図 14.2 のように互い違いに対角上にある頂点における目的関数の値をとる．そこで，各超短形において対角上にある 2 つの頂点から超短形上で取り得る最小値を計算する必要がある．目的関数が Lipschitz 定数 K を持つ場合，

$$f(\boldsymbol{a}) - f(\boldsymbol{x}^*) \leq K\|\boldsymbol{a} - \boldsymbol{x}^*\|_2 \tag{14.5}$$

$$f(\boldsymbol{b}) - f(\boldsymbol{x}^*) \leq K\|\boldsymbol{b} - \boldsymbol{x}^*\|_2 \tag{14.6}$$

となる．ここで，$\boldsymbol{a}, \boldsymbol{b}$ は対角上にある頂点，$\boldsymbol{x}^*$ は超短形上で目的関数が最小となる点である．これらに加えて，

$$\max_{\boldsymbol{x}^* \in \text{超短形}} (\|\boldsymbol{a} - \boldsymbol{x}^*\|_2 + \|\boldsymbol{b} - \boldsymbol{x}^*\|_2) \leq \sqrt{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2 \tag{14.7}$$

が成り立つこと [23, Lemma 2.] を用いると，$\tilde{K} \geq \sqrt{2}K$ となる $\tilde{K}$ について

$$f(\boldsymbol{x}^*) \geq \frac{f(\boldsymbol{a}) + f(\boldsymbol{b})}{2} - \tilde{K}\frac{\|\boldsymbol{a} - \boldsymbol{b}\|_2}{2} \tag{14.8}$$

が成り立つことを示せる [20, Theorem 2.1]．この式の右辺を式 (14.2) の両辺の代わりに用いることで，Lipschitz 定数が $\tilde{K}$ の場合に最小値を取り得る超短形を探索できる．

第 15 章

数値実験

本章では，本書で示した最適化アルゴリズムについて数値実験を行った結果を示す．

15.1　対象としたアルゴリズム

本章において結果を示す図の中で用いる略称とそれに対応する解法を以下に示す．

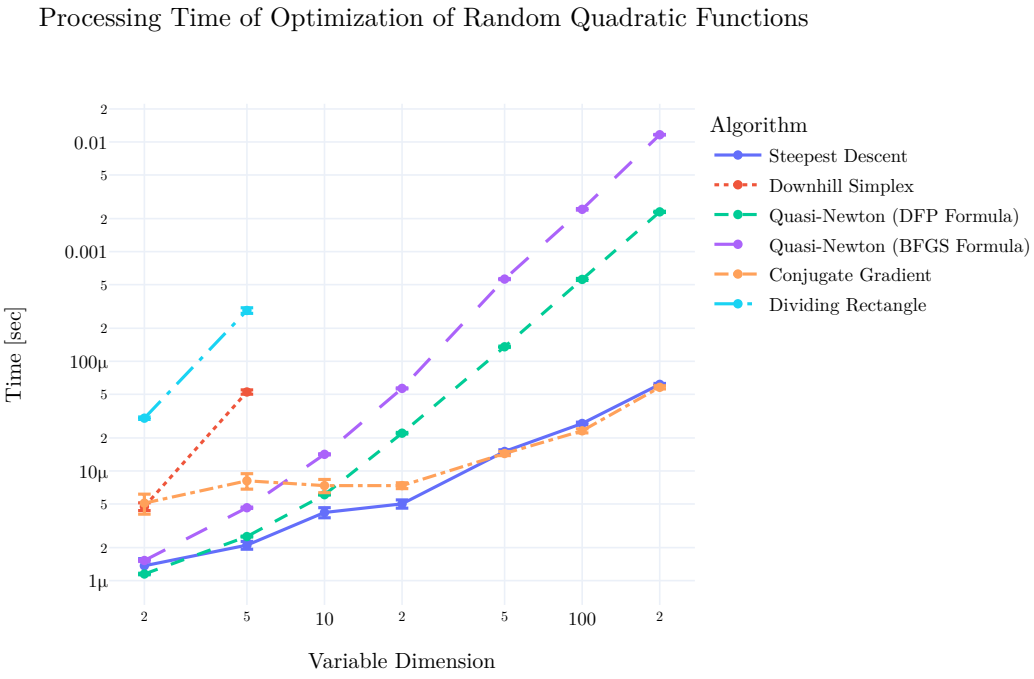
Steepest Descent　最急降下法（12.2 節）
Downhill Simplex　Downhill Simplex 法（13 章）
Quasi-Newton (DFP Formula)　DFP 公式による準 Newton 法（12.4 節）
Quasi-Newton (BFGS Formula)　BFGS 公式による準 Newton 法（12.4 節）
Conjugate Gradient　共役勾配法（12.5 節）
Dividing Rectangle　Dividing Rectangles (DIRECT) 法（14.1 節）
Adaptive Diagonal Curves　Adaptive Diagonal Curves 法（14.2 節）
Gaussian Process Optimization　Gaussian Process 最適化（33.8 節）

15.2　制約のない凸関数の最適化

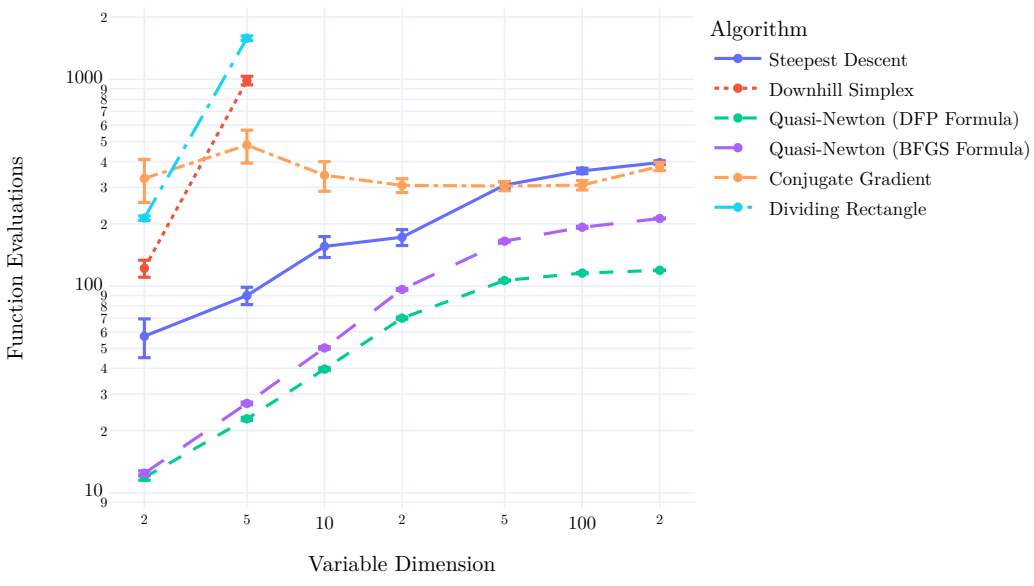
本節では，大域最適解以外に局所最適解をもたない凸関数を制約条件なしで最適化する基本的な問題について数値実験を行った結果を示す．

■ランダムな二次関数の最適化　ランダムな二次関数に最適化アルゴリズムを適用した結果を図 15.1 に示す. Downhill Simplex 法と Dividing Rectangle 法は変数の次元が増加すると急激に計算時間が増加した．そのためこれら 2 つのアルゴリズムについては 5 次元までしか計測を行っていない．二次関数のような単純な関数では単純な最急降下法で十分速く計算ができたようだ．また，共役勾配法では次元が増加しても計算時間が大きく増加していない．

■テスト用の凸関数の最適化　Web ページ [24] で紹介されていたテスト用の凸関数 Rosenbrock Function, Powell Function（4 次元）について最適化アルゴリズムを適用した結果を図 15.2 に示す．これらの関数においては単純な最急降下法の収束が遅くなり，準 Newton 法の方が 2 桁速くなっている．Downhill Simplex 法は中間的な性能となっている．勾配の計算すら必要のないアルゴリズムだが，変数の次元が少ない問題であれば最適解を求めることができている．

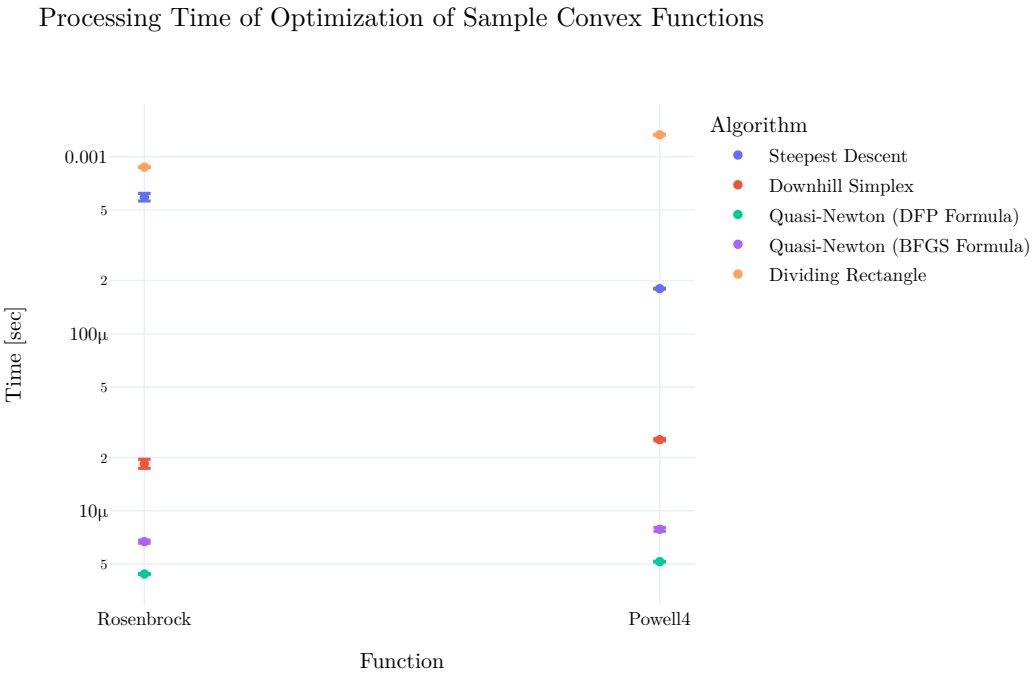


(a) 計算時間

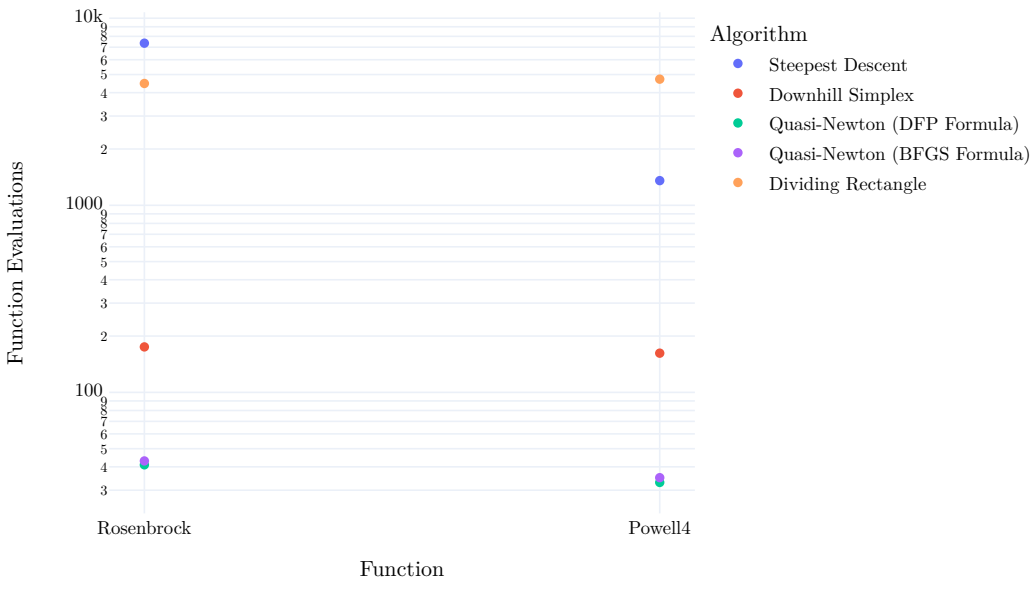


(b) 関数の値を評価した回数

図 15.1: ランダムな二次関数を最適化した場合の計算時間と関数の値を評価した回数



(a) 計算時間



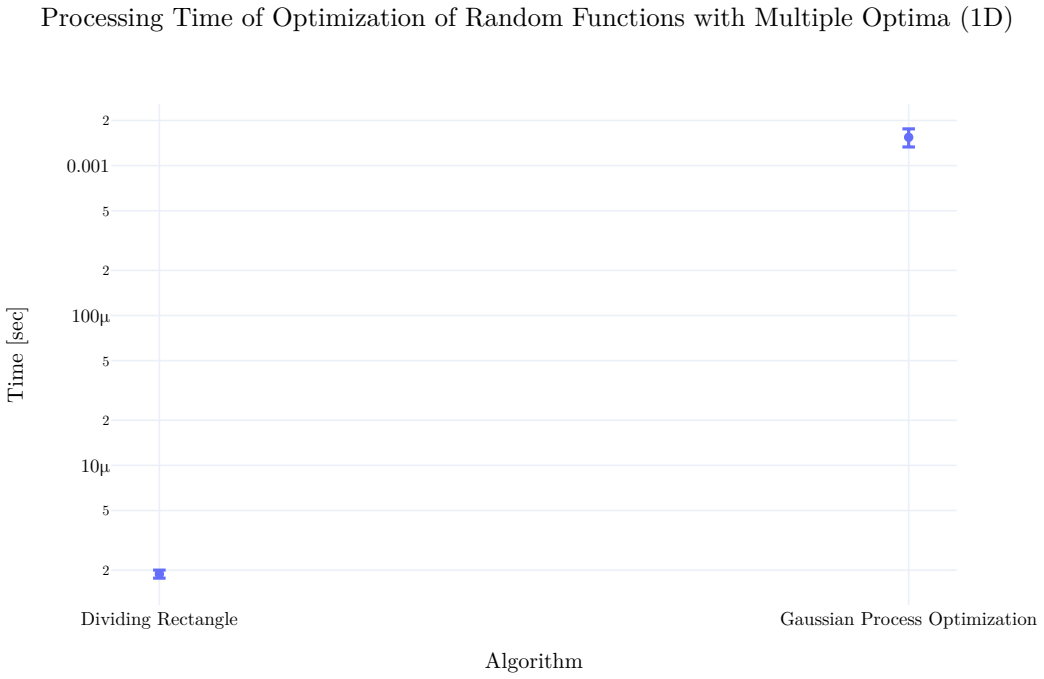
(b) 関数の値を評価した回数

図 15.2: テスト用の凸関数を最適化した場合の計算時間と関数の値を評価した回数

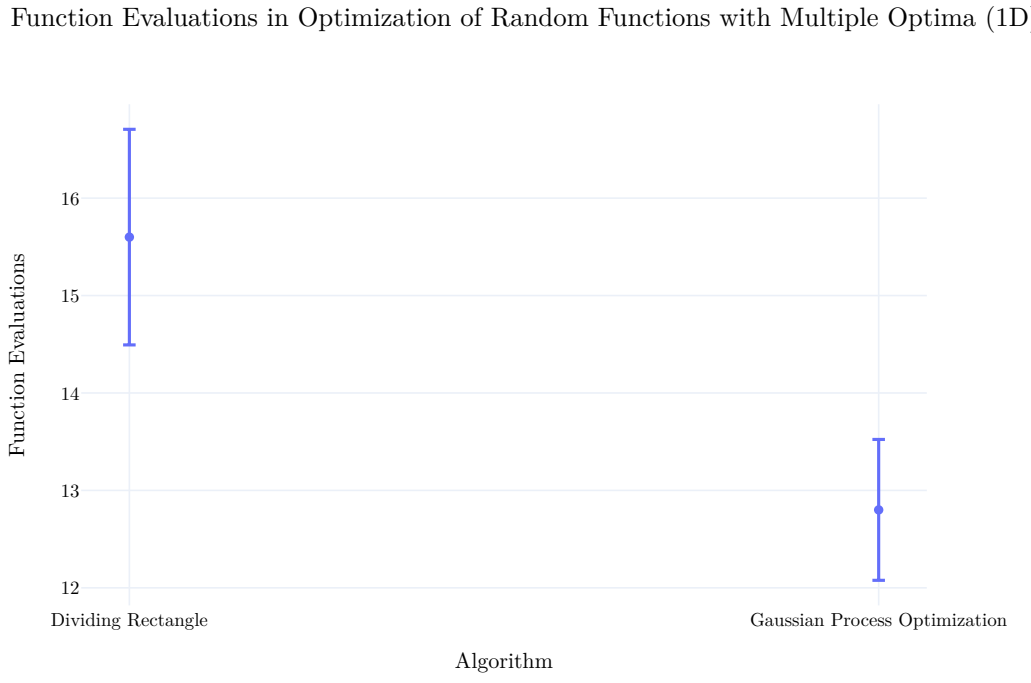
15.3　制約のない大域的最適化

■1 次元における局所最適解を持つランダムな関数の最適化　1 次元において局所最適解を持つ関数をランダムに生成し，最適化アルゴリズムを適用した結果を図 15.3 に示す．計算時間の短い関数は関数の値を評価した回数が多くなっている．目的関数の計算時間が短い場合はアルゴリズム自体の計算時間が短い Dividing Rectangle 法を用いた方が良い一方，目的関数の計算時間が長い場合は関数の値を評価する回数が少なく済む Gaussian Process 最適化を用いた方が良いと思われる．

■複数次元における局所最適解を持つランダムな関数の最適化　複数次元において局所最適解を持つ関数をランダムに生成し，最適化アルゴリズムを適用した結果を図 15.4 に示す．1 次元の場合と同様に計算時間の短いアルゴリズムと関数の評価を行う回数が少ないアルゴリズムが異なっているため，対象とする目的関数の計算時間や難易度に応じてアルゴリズムを切り替えることでより速く解を得られる可能性がある．特に，Adaptive Diagonal Curves は困難な大域的最適化の問題において特に良い性能が出せるという [20, 22]．

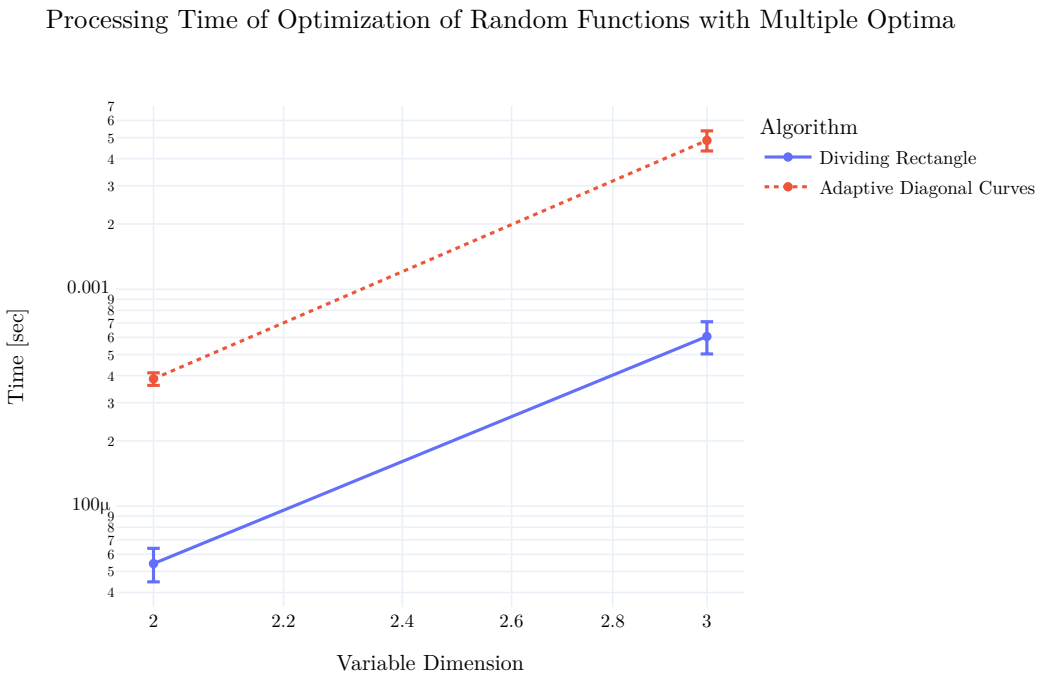


(a) 計算時間

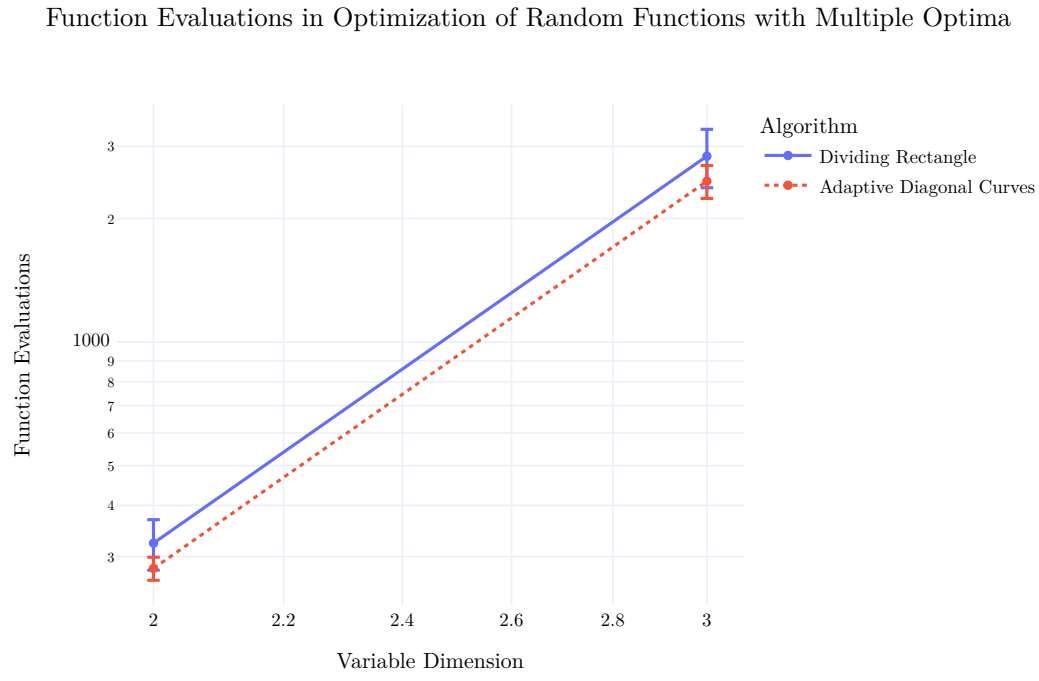


(b) 関数の値を評価した回数

図 15.3: 1 次元における局所最適解を持つランダムな関数を最適化した場合の計算時間と関数の値を評価した回数



(a) 計算時間



(b) 関数の値を評価した回数

図 15.4: 2, 3 次元における局所最適解を持つランダムな関数を最適化した場合の計算時間と関数の値を評価した回数

第 III 部

求根アルゴリズム

第 16 章

導入

求根アルゴリズム (root-finding algorithm) は，微分などの演算を含まない通常の方程式 $f(\boldsymbol{x}) = \mathbf{0}$ $(f : \mathbb{R}^n \to \mathbb{R}^m)$ の解を求めるためのアルゴリズムである．関数 f の種類に依り様々な手法が存在する．

第 17 章

Newton-Raphson 法

Newton-Raphson 法（または単に「Newton 法」とも呼ばれる）は，次の式のような更新式で反復的に方程式 $f(x) = 0$ の根を求める手法である．

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{17.1}$$

17.1　1 次元の方程式の場合

1 次元の方程式 $f(x) = 0$ $(f : \mathbb{R} \to \mathbb{R})$ の場合，1 次近似

$$f(x_{n+1}) \approx f(x_n) + f'(x_n)(x_{n+1} - x_n) \tag{17.2}$$

の根を求めると

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{17.3}$$

と更新式が導かれる．

17.1.1　平方根の算出

ここでは，平方根の算出に Newton-Raphson 法を適用してみる．
$a > 0$ について $\sqrt{a}$ を算出する場合，次の関数 f の正の根を求めれば良い．

$$f(x) = x^2 - a \tag{17.4}$$

この関数を微分すると

$$f'(x) = 2x \tag{17.5}$$

となる．よって，更新式は次のようになる．

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{x_n^2 - a}{2x_n} \\ &= \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \end{aligned} \tag{17.6}$$

定理 17.1. 式 (17.6) の更新式は，初期値 x_0 が $x_0 > 0$ を満たす場合，平方根 $\sqrt{a}$ に収束する．

証明. $x_0 = \sqrt{a}$ の場合は $k = 1, 2, \ldots$ において $x_k = \sqrt{a}$ が成り立つ．つまり，平方根 $\sqrt{a}$ に収束している．そこで，以下では $x_0 \neq \sqrt{a}$ とする．

更新後の値 x_{n+1} と平方根 $\sqrt{a}$ の差は次のようになる．

$$\begin{aligned} & x_{n+1} - \sqrt{a} \\ =& \frac{1}{2}(x_n - \sqrt{a}) + \frac{1}{2}\frac{a - \sqrt{a}x_n}{x_n} \\ =& \frac{1}{2}(x_n - \sqrt{a})\left(1 - \frac{\sqrt{a}}{x_n}\right) \\ =& \frac{1}{2x_n}(x_n - \sqrt{a})^2 \end{aligned} \tag{17.7}$$

$x_0 > 0$ かつ $x_0 \neq \sqrt{a}$ の場合，式 (17.7) より $x_1 - \sqrt{a} > 0$ が成り立つ．さらに，$k = 2, 3, \ldots$ において $x_k - \sqrt{a} > 0$ であることも帰納的に成り立つ．よって，$k = 1, 2, \ldots$ において $x_k - \sqrt{a} > 0$ である．

また，$a > 0$ より $\sqrt{a} > 0$ であることを用いると，

$$\begin{aligned} & x_{n+1} - \sqrt{a} \\ =& \frac{1}{2x_n}(x_n - \sqrt{a})^2 \\ =& \frac{x_n - \sqrt{a}}{2x_n}(x_n - \sqrt{a}) \\ <& \frac{x_n}{2x_n}(x_n - \sqrt{a}) \\ =& \frac{1}{2}(x_n - \sqrt{a}) \end{aligned} \tag{17.8}$$

となる．

以上から，

$$0 < x_{n+1} - \sqrt{a} < \frac{1}{2}(x_n - \sqrt{a}) \tag{17.9}$$

が成り立ち，x_n の列が平方根 $\sqrt{a}$ へ収束する． □

17.1.2 べき根の算出

一般に r 乗根（$r = 2, 3, \ldots$）を算出することを考える．

$a > 0$ について $\sqrt[r]{a}$ を算出する場合，次の関数 f の正の根を求めれば良い．

$$f(x) = x^r - a \tag{17.10}$$

この関数を微分すると

$$f'(x) = rx^{r-1} \tag{17.11}$$

となる．よって，更新式は次のようになる．

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{x_n^r - a}{rx_n^{r-1}} \\ &= \frac{r-1}{r}x_n + \frac{a}{rx_n^{r-1}} \end{aligned} \tag{17.12}$$

定理 17.2. 式 (17.12) の更新式は，初期値 x_0 が $x_0 > 0$ を満たす場合，べき根 $\sqrt[r]{a}$ に収束する．

証明. $x_0 = \sqrt[r]{a}$ の場合は $k = 1, 2, \ldots$ において $x_k = \sqrt[r]{a}$ が成り立つ．つまり，平方根 $\sqrt[r]{a}$ に収束している．そこで，以下では $x_0 \neq \sqrt[r]{a}$ とする．

更新後の値 x_{n+1} とべき根 $\sqrt[r]{a}$ の差は次のようになる．

$$\begin{aligned} & x_{n+1} - \sqrt[r]{a} \\ =& \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) + \frac{1}{r}\left(\frac{a}{x_n^{r-1}} - \sqrt[r]{a}\right) \\ =& \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) - \frac{\sqrt[r]{a}}{r}\left(1 - \left(\frac{\sqrt[r]{a}}{x_n}\right)^{r-1}\right) \end{aligned} \tag{17.13}$$

これに等比級数の公式

$$\sum_{k=1}^{n} x^{k-1} = \frac{1-x^n}{1-x} \tag{17.14}$$

を適用すると，

$$\begin{aligned} & x_{n+1} - \sqrt[r]{a} \\ =& \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) - \frac{\sqrt[r]{a}}{r}\left(1 - \frac{\sqrt[r]{a}}{x_n}\right)\sum_{k=1}^{r-1}\left(\frac{\sqrt[r]{a}}{x_n}\right)^{k-1} \\ =& \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) - \frac{1}{r}\left(x_n - \sqrt[r]{a}\right)\sum_{k=1}^{r-1}\left(\frac{\sqrt[r]{a}}{x_n}\right)^{k} \\ =& \frac{1}{r}\left(x_n - \sqrt[r]{a}\right)\left(r - 1 - \sum_{k=1}^{r-1}\left(\frac{\sqrt[r]{a}}{x_n}\right)^{k}\right) \qquad (17.15) \\ =& \frac{1}{r}\left(x_n - \sqrt[r]{a}\right)\sum_{k=1}^{r-1}\left(1 - \left(\frac{\sqrt[r]{a}}{x_n}\right)^{k}\right) \\ =& \frac{1}{r}\left(x_n - \sqrt[r]{a}\right)\left(1 - \frac{\sqrt[r]{a}}{x_n}\right)\sum_{k=1}^{r-1}\sum_{l=1}^{k}\left(\frac{\sqrt[r]{a}}{x_n}\right)^{l-1} \\ =& \frac{1}{rx_n}\left(x_n - \sqrt[r]{a}\right)^2\sum_{k=1}^{r-1}\sum_{l=1}^{k}\left(\frac{\sqrt[r]{a}}{x_n}\right)^{l-1} \qquad (17.16) \end{aligned}$$

となる．

$x_0 > 0$ かつ $x_0 \neq \sqrt[r]{a}$ の場合，式 (17.16) より $x_1 - \sqrt[r]{a} > 0$ が成り立つ．さらに，$k = 2, 3, \ldots$ において $x_k - \sqrt[r]{a} > 0$ であることも帰納的に成り立つ．よって，$k = 1, 2, \ldots$ において $x_k - \sqrt[r]{a} > 0$ である．

さらに，$x_n > \sqrt[r]{a} > 0$ であることから $\sqrt[r]{a}/x_n > 0$ が成り立つから，式 (17.15) より

$$\begin{aligned} & x_{n+1} - \sqrt[r]{a} \\ <& \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) \end{aligned} \tag{17.17}$$

となる．

以上から，

$$0 < x_{n+1} - \sqrt[r]{a} < \frac{r-1}{r}\left(x_n - \sqrt[r]{a}\right) \tag{17.18}$$

が成り立ち，x_n の列がべき根 $\sqrt[r]{a}$ へ収束する． □

r が奇数の場合，負の数 $a < 0$ に対してべき根 $\sqrt[r]{a} < 0$ が存在する．この場合についてもべき根の算出を考える．

定理 17.3. $r = 3, 5, 7, \ldots$ かつ $a < 0$ の場合を考える．式 (17.12) の更新式は，初期値 x_0 が $x_0 < 0$ を満たす場合，べき根 $\sqrt[r]{a}$ に収束する．

証明. $x_0 = \sqrt[r]{a}$ の場合は $k = 1, 2, \ldots$ において $x_k = \sqrt[r]{a}$ が成り立つ．つまり，平方根 $\sqrt[r]{a}$ に収束している．そこで，以下では $x_0 \neq \sqrt[r]{a}$ とする．

$x_0 < 0$ かつ $x_0 \neq \sqrt[r]{a}$ の場合，式 (17.16) より $x_1 - \sqrt[r]{a} < 0$ が成り立つ．さらに，$k = 2, 3, \ldots$ において $x_k - \sqrt[r]{a} < 0$ であることも帰納的に成り立つ．よって，$k = 1, 2, \ldots$ において $x_k - \sqrt[r]{a} < 0$ である．

さらに，$x_n < \sqrt[r]{a} < 0$ であることから $\sqrt[r]{a}/x_n > 0$ が成り立つから，式 (17.15) より

$$\begin{aligned} & x_{n+1} - \sqrt[r]{a} \\ >& \frac{r-1}{r} \left(x_n - \sqrt[r]{a} \right) \end{aligned} \tag{17.19}$$

となる．

以上から，

$$\frac{r-1}{r} \left(x_n - \sqrt[r]{a} \right) < x_{n+1} - \sqrt[r]{a} < 0 \tag{17.20}$$

が成り立ち，x_n の列がべき根 $\sqrt[r]{a}$ へ収束する． □

17.2 一般の次元の方程式の場合

一般に n 次元（$n = 1, 2, \ldots$）の方程式 $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ $(\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n)$ の場合，1 次近似

$$\boldsymbol{f}(\boldsymbol{x}_{n+1}) \approx \boldsymbol{f}(\boldsymbol{x}_n) + \left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_n} (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n) \tag{17.21}$$

をもとに更新式

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \left(\left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_n} \right)^{-1} \boldsymbol{f}(\boldsymbol{x}_n) \tag{17.22}$$

が得られる．

17.3 減速 Newton 法

Newton-Raphson 法の更新式 (17.22) は 1 次近似により導出しているため，近似によるずれが大きい場合は更新により方程式の解から離れてしまう可能性がある．そのような場合に対応するため，ステップ幅 $t_n > 0$ を加えた更新式

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - t_n \left(\left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\boldsymbol{x}_n} \right)^{-1} \boldsymbol{f}(\boldsymbol{x}_n) \tag{17.23}$$

による減速 Newton 法が考えられている [25, 6.6 節]．ステップ幅 t_n の決定法については，以下のようなものが考えられている．

- $\boldsymbol{x}_n$ に関係ない $t_n = t$ に固定する [25, 6.6 節].
- 動的に決定する．

- $\|\boldsymbol{f}(\boldsymbol{x}_{n+1})\|_2 < \|\boldsymbol{f}(\boldsymbol{x}_n)\|_2$ となるまでステップ幅を縮小する [25, 6.6 節].
- 最適化の直線探索を適用する（17.4 節参照）.

これにより，ステップ幅のない更新式よりも数値解を求められる問題の幅が広がる.

17.4 最適化との関係

Newton-Raphson 法の更新式 (17.22) は，最適化における勾配法（12 章）の枠組みでとらえることができる．そのことを利用すると，Backtracking Line Search (Algorithm 12.2) のような直線探索の手法を Newton-Raphson 法にも取り入れることができる.

n 次元（$n = 1, 2, \ldots$）の方程式 $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ ($\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$) に対して最適化の目的関数

$$g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|_2^2 \tag{17.24}$$

を考える．このとき，勾配は

$$\nabla g(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^\top \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \tag{17.25}$$

となるため，Newton-Raphson 法の更新式における更新方向

$$\boldsymbol{d} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1} \boldsymbol{f}(\boldsymbol{x}) \tag{17.26}$$

は，$\boldsymbol{f}(\boldsymbol{x}) = 0$ でない限り

$$\begin{aligned} \nabla g(\boldsymbol{x})^\top \boldsymbol{d} &= -\boldsymbol{f}(\boldsymbol{x})^\top \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1} \boldsymbol{f}(\boldsymbol{x}) \\ &= -\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{f}(\boldsymbol{x}) \\ &= -\|\boldsymbol{f}(\boldsymbol{x})\|_2^2 \\ &< 0 \end{aligned} \tag{17.27}$$

のように目的関数の降下方向となっている.

以上から，目的関数 $g(\boldsymbol{x})$ の最適化を行うものとして最適化の直線探索手法 (12.1 節) を適用することができる.

第 IV 部

常微分方程式の数値解法

第 18 章

導入

この部では，常微分方程式 (ordinary differential equation, ODE) の数値解法をまとめる．
関数 $\boldsymbol{y}(t)$ の常微分方程式は一般に

$$\boldsymbol{f}(t, \boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}, \dddot{\boldsymbol{y}}, \ddddot{\boldsymbol{y}}, \ldots) = \boldsymbol{0} \tag{18.1}$$

のように書ける．

第 19 章

Runge-Kutta 法

Runge-Kutta 法 (Runge-Kutta method) では次のような形式の初期値問題を数値的に解く [26].

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}) \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases} \tag{19.1}$$

Runge-Kutta 法は，時刻 t における変数 $\boldsymbol{y}(t)$ の値から，次のような形式で時刻 $t+h$ における変数 $\boldsymbol{y}(t+h)$ の計算を行う．

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j\right) \qquad \text{for } i = 1, 2, \ldots, s \tag{19.2}$$

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + h \sum_{i=1}^{s} c_i \boldsymbol{k}_i \tag{19.3}$$

ここで，時間の更新幅 h はステップ幅と呼ばれる．Runge-Kutta 法には，整数 s（段数と呼ばれる）と係数 a_{ij}, b_i, c_i の異なる様々な公式が存在する．段数が増えるにつれて数を増していく係数 a_{ij}, b_i, c_i は表 19.1 のような Butcher 配列と呼ばれる形式で記載されることが多い．

Runge-Kutta 法の公式は次のように分類される [26].

陽的 Runge-Kutta 法 (Explicit Runge-Kutta method, ERK method)　$j \geq i$ について $a_{ij} = 0$ となっている場合，$\boldsymbol{ki}$ は $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_s$ の順に式 (19.2) の右辺を評価することで計算できる．このような場合は陽的 Runge-Kutta 法と呼ばれる．

半陰的 Runge-Kutta 法 (Semi-implicit Runge-Kutta method)　$j > i$ について $a_{ij} = 0$ となっている場合，$\boldsymbol{ki}$ は $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_s$ の順に式 (19.2) を $\boldsymbol{k}_i$ について解くことで計算できる．このような場合は半陰的 Runge-Kutta 法と呼ばれる．[*1]

表 19.1: Butcher 配列における係数の並べ方

b_1	a_{11}	a_{12}	a_{13}	$\cdots$	a_{1s}
b_2	a_{21}	a_{22}	a_{23}	$\cdots$	a_{2s}
b_3	a_{31}	a_{32}	a_{33}	$\cdots$	a_{3s}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
b_s	a_{s1}	a_{s2}	a_{s3}	$\cdots$	a_{ss}
	c_1	c_2	c_3	$\cdots$	c_s

[*1] Diagonally implicit Runge-Kutta (DIRK) とも呼ばれる [27].

陰的 Runge-Kutta 法 (Implicit Runge-Kutta method, IRK method) $j > i$ でも $a_{ij} \neq 0$ となる係数が存在する場合，$\boldsymbol{ki}$ は $i = 1, 2, \ldots, s$ について連立した式 (19.2) を $\boldsymbol{k}_i$ について解くことで計算する．このような場合は陰的 Runge-Kutta 法と呼ばれる．[*2]

陽的 Runge-Kutta 法の方が計算は単純だが，陰的 Runge-Kutta 法は

- 硬い系と呼ばれる比較的不安定な微分方程式で解が安定しやすい．
- 陽的 Runge-Kutta 法よりも少ない段数でより高い次数を出せる．（後述する公式の実例を見ると分かる．）

といった利点を持つ．そのため，解が安定するようにステップ幅を調整した場合，陰的 Runge-Kutta 法の方がステップ幅を大きくとることができ，目的の時刻 t までの解を得るために必要な計算時間は少なくなる場合がある．

また，Runge-Kutta 法の公式の精度を示す数値として次数が存在する．変数の近似値 $\boldsymbol{y}(t+h)$ が厳密解の Tailor 展開が p じまで一致する場合，その公式は p 次といい，p は次数と呼ばれる．変数の近似値 $\boldsymbol{y}(t+h)$ の精度は h^{p+1} オーダーとなる．

19.1 埋め込み型公式

Runge-Kutta 法の公式の中には，複数の c_i の組を持つものがある（表 19.3 の例を参照）．そのような公式では，次のような 2 つの近似値を得ることができる．

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + \sum_{i=1}^{s} c_i \boldsymbol{k}_i \tag{19.4}$$

$$\boldsymbol{y}^*(t+h) = \boldsymbol{y}^*(t) + \sum_{i=1}^{s} c_i^* \boldsymbol{k}_i \tag{19.5}$$

これらの差により誤差の近似値を求めることができる．

$$\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h) = \sum_{i=1}^{s} (c_i - c_i^*) \boldsymbol{k}_i \tag{19.6}$$

2 つの異なる公式を用いることによっても同じことはできるが，埋め込み型の公式の方が $\boldsymbol{k}_i$ を共有できて効率が良い．

19.1.1 ステップ幅の自動更新

埋め込み型公式を用いると，次のように現在のステップ幅から次のステップ幅 $\hat{h}$ の適正値を推定できる [26, 4.1 節 (a)]．まず，c_i と c_i^* のうち次数が低い方の次数を p とすると，

$$|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)| \approx |Ah^{p+1}| \tag{19.7}$$

のように書ける．そこで，誤差の許容量を ε_{tol} としたとき，次の方程式が成り立つようにする．

$$\frac{\hat{h}^{p+1}}{h^{p+1}} = \frac{\varepsilon_{tol}}{|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)|} \tag{19.8}$$

これを $\hat{h}$ について解くと，次のようになる．

$$\hat{h} = h \left(\frac{\varepsilon_{tol}}{|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)|} \right)^{\frac{1}{p+1}} \tag{19.9}$$

[*2] 半陰的 Runge-Kutta 法まで含めて陰的 Runge-Kutta 法と呼ばれていることもある．

Algorithm 19.1 式 (19.9) を利用したステップサイズの自動決定 [26, 4.1 節 (a)], [28, Section II.4]

procedure StepWithAutoStepSize($\boldsymbol{f}, t, \boldsymbol{y}(t), h$)
  loop
    現在のステップ幅 h で 1 ステップ計算し，推定値 $\boldsymbol{y}(t+h)$, $\boldsymbol{y}^*(t+h)$ を計算する
    if $|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)| < \varepsilon_{tol}$ **then**
      $f_h \leftarrow (\varepsilon_{tol}/|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)|)^{\frac{1}{p+1}}$ ▷ 式 (19.9) における倍率
      $f_h \leftarrow f_{safe} f_h$ ▷ f_{safe} は安全係数 [26, 4.1 節 (a)], [28, Section II.4]
      if $f_h > f_{max}$ **then** ▷ 小さい推定誤差によりステップ幅が大きくなりすぎることを防ぐ [28, Section II.4]
        $f_h \leftarrow f_{max}$
      end if
      if $f_h < f_{min}$ **then** ▷ 倍率に下限を持たせることも有用 [28, Section II.4]
        $f_h \leftarrow f_{min}$
      end if
      $h \leftarrow f_h h$
      return h, $\boldsymbol{y}(t+h)$
    end if
    $h \leftarrow rh$ ▷ 比率 $r \in (0,1)$ でステップサイズを縮小する
  end loop
end procedure

以上を元に，Algorithm 19.1 のように自動でステップサイズを決定することができる．Algorithm 19.1 における係数 f_{safe} は安全係数で $f_{safe} = 0.8, 0.9, (0.25)^{1/(p+1)}, (0.38)^{1/(p+1)}$ といった値が用いられ，ステップ幅の倍率の最大値 f_{max} は 1.5 から 5 までの値が用いられる [28, Section II.4]．なお，解のベクトルの各次元について誤差の許容量を指定する場合，解の次元を d として

$$err = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\frac{[\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)]_i}{[\varepsilon_{tol}]_i}\right)^2} \tag{19.10}$$

のように平均をとり，式 (19.9) の代わりに

$$\hat{h} = h\left(\frac{1}{err}\right)^{\frac{1}{p+1}} \tag{19.11}$$

を用いる方法も考えられている [28, Section II.4]．

19.1.2 PI 制御によるステップ幅の自動更新

ステップ幅を更新する方法として，前節の式 (19.11) を制御に用いられる PI 制御により改良した手法が考案されている [27, Section IV.2]．

n 回目の反復におけるステップ幅を h_n，式 (19.10) の相対的な推定誤差の比率を err_n とする．そこで，入力

Algorithm 19.2 簡易的なステップサイズの自動決定 [26, 4.1 節 (a)]

procedure StepWithAutoStepSize($\boldsymbol{f}, t, \boldsymbol{y}(t), h$)
  loop
    現在のステップ幅 h で 1 ステップ計算し，推定値 $\boldsymbol{y}(t+h)$, $\boldsymbol{y}^*(t+h)$ を計算する
    if $|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)| < \varepsilon_{tol}$ **then**
      if $|\boldsymbol{y}(t+h) - \boldsymbol{y}^*(t+h)| < \frac{1}{2^{p+1}}\varepsilon_{tol}$ **then** ▷ ステップサイズを 2 倍にしても許容誤差の範囲内になる
        $h \leftarrow 2h$
      end if
      return h, $\boldsymbol{y}(t+h)$
    end if
    $h \leftarrow rh$ ▷ 比率 $r \in (0,1)$ でステップサイズを縮小する
  end loop
end procedure

$C = \log h_n$ を適切に変更することにより出力 $\theta = \log err_n$ を制御することを考える[*3]．このとき，PI 制御の式[*4]

$$-\dot{C} = n_1\theta + n_2\dot{\theta} \tag{19.12}$$

に対して次のような近似が考えられる．

$$-(\log h_{n+1} - \log h_n) = n_1 \log err_n + n_2(\log err_n - \log err_{n-1}) \tag{19.13}$$

$\alpha = n_1 + n_2$, $\beta = n_2$ として式を整理すると次のようになる．

$$h_{n+1} = h_n \frac{(err_{n-1})^\beta}{(err_n)^\alpha} \tag{19.14}$$

これを用いて Algorithm 19.1 のようにステップ幅の自動更新を行う．パラメータ α, β には次のような値が用いられる．

- $\alpha = 1/(p+1)$, $\beta = 0.08$. [27, Section IV.2]
- $\alpha = 0.7/(p+1)$, $\beta = 0.4/(p+1)$. [29]

19.1.3 ステップ幅の簡易的な自動更新

推定誤差をもとにしたステップ幅の更新方法としては，単純化した Algorithm 19.2 のような手法も存在する．

19.1.4 初回のステップ幅の自動決定

ここまでに示した方法では何らかのステップ幅が与えられている状態を前提としているが，初期値問題における最初のステップでは人間が適当なステップ幅を与えない限りステップ幅が存在しない状態から始めることになる．そこで，初回のステップ幅を自動決定する方法について文献 [28, Section II.4] で示されている方法を Algorithm 19.3 に示しておく．

[*3] 前節と表記の整合を取るため，元の文献 [27, Section IV.2] とは少し異なる式になっている．
[*4] 右辺第一項は積分項（Integral feedback），右辺第二項は比例項（Proportional feedback）であり，2 つの頭文字から PI 制御と呼ばれる．

Algorithm 19.3 初期ステップサイズの自動決定 [28, Section II.4]

procedure DETERMINEINITIALSTEPSIZE($\boldsymbol{f}, \boldsymbol{y}(0)$)
  $d_0 \leftarrow \|\boldsymbol{y}_0\|_{tol}$ ▷ ノルム $\|\cdot\|_{tol}$ は式 (19.10) によるもの
  $d_1 \leftarrow \|\boldsymbol{f}(0, \boldsymbol{y}_0)\|_{tol}$
  if $d_0 \geq 10^{-5}$ and $d_1 \geq 10^{-5}$ **then**
    $h_0 \leftarrow 0.01 d_0/d_1$ ▷ 陽的 Euler 法（19.4.3 節）における変化量を小さくする
  else
    $h_0 \leftarrow 10^{-6}$
  end if
  $\boldsymbol{y}_1 \leftarrow \boldsymbol{y}_0 + h_0 \boldsymbol{f}(0, \boldsymbol{y}_0)$ ▷ 陽的 Euler 法の 1 ステップ分の計算を行う
  $d_2 \leftarrow \|\boldsymbol{f}(h_0, \boldsymbol{y}_1) - \boldsymbol{f}(0, \boldsymbol{y}_0)\|_{tol}/h_0$ ▷ 2 階導関数の評価値
  if $\max\{d_1, d_2\} > 10^{-15}$ **then**
    $h_1 \leftarrow \sqrt[p+1]{0.01/\max\{d_1, d_2\}}$
  else
    $h_1 \leftarrow \max\{10^{-6}, 10^{-3} h_0\}$
  end if
  $h \leftarrow \min\{100 h_0, h_1\}$
end procedure

19.2 埋め込み型でない公式による誤差の評価

埋め込み型でない公式でも式 (19.7) のように誤差の評価を行い Algorithm 19.1 のようにステップサイズを自動決定を行う方法が考えられており，Milne's device [26, 4.1 節 (b)] と呼ばれる．

次のように Runge-Kutta 法を適用する演算子を考える．

$$F[\boldsymbol{f}, t, \boldsymbol{y}(t), h] \equiv \boldsymbol{y}(t) + \sum_{i=1}^{s} c_i \boldsymbol{k}_i \tag{19.15}$$

使用した公式の次数が p であれば，厳密解 $\overline{\boldsymbol{y}(t+h)}$ との誤差は

$$F[\boldsymbol{f}, t, \boldsymbol{y}(t), h] - \overline{\boldsymbol{y}(t+h)} = Ah^{p+1} + O(h^{p+2}) \tag{19.16}$$

のようになる．ステップサイズを半分にして 2 ステップ解くと

$$F\left[\boldsymbol{f}, t, \boldsymbol{y}(t), \frac{1}{2}h\right] - \overline{\boldsymbol{y}\left(t + \frac{1}{2}h\right)} = \frac{1}{2^{p+1}} Ah^{p+1} + O(h^{p+2}) \tag{19.17}$$

$$F\left[\boldsymbol{f}, t + \frac{h}{2}, F\left[\boldsymbol{f}, t, \boldsymbol{y}(t), \frac{1}{2}h\right], \frac{1}{2}h\right] - \overline{\boldsymbol{y}(t+h)} = \frac{1}{2^{p+1}}(A + A')h^{p+1} + O(h^{p+2}) \tag{19.18}$$

のようになる．よって，

$$F[\boldsymbol{f}, t, \boldsymbol{y}(t), h] - F\left[\boldsymbol{f}, t + \frac{h}{2}, F\left[\boldsymbol{f}, t, \boldsymbol{y}(t), \frac{1}{2}h\right], \frac{1}{2}h\right] = \left(A + \frac{1}{2^{p+1}}(A + A')\right) h^{p+1} \tag{19.19}$$

となる．これを用いると，埋め込み型の公式と同様に Algorithm 19.1 を用いることができる．

19.3 陰的 Runge-Kutta 法における方程式の解法

陰的 Runge-Kutta 法においては，式 (19.2) を方程式として解く必要がある．数値解法に頼りたくなるような状況下において関数 $\boldsymbol{f}$ は非線形であるため[*5]，Newton-Raphson 法（17 章）のような非線形方程式の数値解法を用いる必要がある．

19.3.1 半陰的公式の場合

半陰的公式の場合，$i = 1, 2, \ldots, s$ のそれぞれについて以下の方程式を解く必要がある．

$$\boldsymbol{F}_i(\boldsymbol{k}_i) \equiv \boldsymbol{k}_i - \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h\sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j\right) = \boldsymbol{0} \tag{19.20}$$

この方程式の数値解法として Newton-Raphson 法を用いる場合，Jacobian

$$\left.\frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{k}_i}\right|_{\boldsymbol{k}_i=\boldsymbol{k}_i} = I - ha_{ii}\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{t=t+b_i h, \boldsymbol{y}=\boldsymbol{y}(t)+h\sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j} \tag{19.21}$$

を使用する必要がある．Newton-Raphson 法における更新式は次のようになる．

$$(\boldsymbol{k}_i)_{n+1} = (\boldsymbol{k}_i)_n - \left(\left.\frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{k}_i}\right|_{\boldsymbol{k}_i=\boldsymbol{k}_i}\right)^{-1} \boldsymbol{F}_i(\boldsymbol{k}_i) \tag{19.22}$$

Newton-Raphson 法の各反復ごとに更新された Jacobian の LU 分解を求める必要がある．

ステップ幅を十分小さくとっている場合，Jacobian の $\boldsymbol{k}_i$ に応じた変化量は少ないため，Jacobian を $\boldsymbol{k}_i = \boldsymbol{0}$ のときの

$$J \equiv I - ha_{ii}\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{t=t+b_i h, \boldsymbol{y}=\boldsymbol{y}(t)} \tag{19.23}$$

に固定するという近似もあり得る [26, 6.2 節 (c)]．この場合，1 段に 1 回だけ Jacobian とその LU 分解を計算すれば良い[*6]．

19.3.2 陰的公式の場合

半陰的公式でなく陰的公式の場合，必要な方程式は次元が増えた次のようなものになる．

$$\boldsymbol{F}_i(\boldsymbol{k}_i) \equiv \begin{pmatrix} \boldsymbol{k}_1 - \boldsymbol{f}\left(t + b_1 h, \boldsymbol{y}(t) + h\sum_{j=1}^{s} a_{1j}\boldsymbol{k}_j\right) \\ \boldsymbol{k}_2 - \boldsymbol{f}\left(t + b_2 h, \boldsymbol{y}(t) + h\sum_{j=1}^{s} a_{2j}\boldsymbol{k}_j\right) \\ \vdots \\ \boldsymbol{k}_s - \boldsymbol{f}\left(t + b_s h, \boldsymbol{y}(t) + h\sum_{j=1}^{s} a_{sj}\boldsymbol{k}_j\right) \end{pmatrix} = \boldsymbol{0} \tag{19.24}$$

[*5] 関数 $\boldsymbol{f}$ が線形な場合，行列の指数関数による一般解が存在し，固有値分解により簡単に解を得ることが可能であるため，あえて Runge-Kutta 法を使用する必要はない．

[*6] d 次元の LU 分解において，係数行列が更新された場合は d^3 オーダーの時間がかかり，係数行列が変わらず適応先のベクトルのみが更新された場合は d^2 オーダーの時間で済む．対象とする常微分方程式の自由度の数が多いほど効果が大きくなる．

Jacobian は次のようになる．

$$\left.\frac{\partial \boldsymbol{F}_i}{\partial(\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_s)}\right|_{\boldsymbol{k}_i=\boldsymbol{k}_i} = I - \begin{pmatrix} J_{11} & J_{12} & \cdots & J_{1s} \\ J_{21} & J_{22} & \cdots & J_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ J_{s1} & J_{s2} & \cdots & J_{ss} \end{pmatrix} \tag{19.25}$$

$$J_{ij} = ha_{ij} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{t=t+c_i h, \boldsymbol{y}=\boldsymbol{y}(t)+h\sum_{l=1}^{i} a_{il}\boldsymbol{k}_l} \tag{19.26}$$

半陰的公式と同様に，上記の Jacobian を $\boldsymbol{k}_i = \boldsymbol{0}$ のときのものに固定することで計算時間を減らすことが考えられる．陰的公式では Jacobian のサイズが大きくなるため，Jacobian を固定することの効果はより大きくなる[*7]．

19.3.3 陰的公式の別の定式化

ここまで示した解法は $i = 1, 2, \ldots, s$ の各段における方程式を

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h\sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j\right) \tag{19.27}$$

としているが，代わりに

$$\boldsymbol{z}_i = h\sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j \tag{19.28}$$

のような変数を導入し，方程式

$$\boldsymbol{z}_i = h\sum_{j=1}^{s} a_{ij}\boldsymbol{f}\left(t + b_j h, \boldsymbol{y}(t) + \boldsymbol{z}_j\right) \tag{19.29}$$

を解くこともできる [27, Section IV.8.]．

この形式の方程式は，$a_{sj} = c_j$ $(j = 1, \ldots, s)$ が成り立つ場合に次のステップの解の計算を

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + \sum_{i=1}^{s} c_i\boldsymbol{k}_i = \boldsymbol{y}(t) + \boldsymbol{z}_s \tag{19.30}$$

のように簡易化できるという利点を持っている．なお，このように $a_{sj} = c_j$ $(j = 1, \ldots, s)$ となる公式は stiffly accurate であると言われる [27, Section IV.6.] [*8]．

19.3.4 Jacobian の計算が困難な場合

問題の次元の数が大きく Jacobian のメモリ容量が大きすぎる場合や，Jacobian の計算式を用意するのが困難な場合など，Jacobian を用いて Newton 法を実装するのが困難な場合がある．そのような場合でも Newton 法で式 (19.20), (19.24), (19.29) のような方程式を解くために後述の GMRES (Generalized Minimal Residual) 法を組み合わせた Newton-GMRES 法が用いられている [30, 31]．

GMRES 法は Algorithm 19.4 のようにして解を計算する．$\boldsymbol{H}_m \in \mathbb{R}^{(m+1)\times m}$ に QR 分解を適用できる程度に小さい m を用いることで，解きたい方程式の次元が大きい場合でも方程式の近似解を計算することができる．さらに，

[*7] d 次元の常微分方程式に対して s 段公式を使用すると，Jacobian は ds 次元の正方行列となる．

[*8] stiffly accurate な公式は，ただ計算が楽になるというだけでなく，追加の条件を満たしていれば L 安定と呼ばれる安定性を満たす [27, Section IV.3.].

Algorithm 19.4 GMRES (Generalized Minimal Residual) [4, 31]

procedure GMRES($A \in \mathbb{R}^{n\times n}, \boldsymbol{b} \in \mathbb{R}^n, \boldsymbol{x}_0 \in \mathbb{R}^n, m \in \{1, 2, \ldots, n\}$)
  $k \leftarrow 0$
  $\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - A\boldsymbol{x}_0$
  $\beta_0 \leftarrow \|\boldsymbol{r}_0\|_2$
  while $\beta_k > 0$ and $k < m$ **do**
    $\boldsymbol{q}_{k+1} \leftarrow \frac{\boldsymbol{r}_k}{\beta_k}$
    $k \leftarrow k + 1$
    $\boldsymbol{r}_k \leftarrow A\boldsymbol{q}_k$
    for $i = 1, k$ **do**
      $h_{ik} \leftarrow \boldsymbol{q}_i^\top \boldsymbol{r}_k$
      $\boldsymbol{r}_k \leftarrow \boldsymbol{r}_k - h_{ik}\boldsymbol{q}_i$
    end for
    $\beta_k \leftarrow \|\boldsymbol{r}_k\|_2$
    $h_{k+1,k} \leftarrow \beta_k$
  end while
  Compute $\boldsymbol{y}_k = \min_{\boldsymbol{y}\in\mathbb{R}^m} \|\beta_0 \boldsymbol{e}_1 - H_k \boldsymbol{y}\|_2$ for $H_k = \{h_{ij}\}_{1\le k+1, 1\le j\le k}$. ▷ $\boldsymbol{e}_1 = (1, 0, 0, \ldots, 0)$
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0 + Q_k \boldsymbol{y}_k$ ▷ $Q_k = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k) \in \mathbb{R}^{n\times k}$
  return $\boldsymbol{x}$
end procedure

GMRES 法を複数回適用することで解の精度を高めていくことができる [4, 31]．ここで問題となるのは $\boldsymbol{r}_k = A\boldsymbol{q}_k$ の計算だが，Newton 法においては A が Jacobian になるため，

$$A\boldsymbol{q}_k = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\boldsymbol{q}_k \approx \frac{\boldsymbol{F}(\boldsymbol{y} + \varepsilon \boldsymbol{q}_k) - \boldsymbol{F}(\boldsymbol{y})}{\varepsilon} \tag{19.31}$$

または，

$$A\boldsymbol{q}_k = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\boldsymbol{q}_k \approx \frac{\boldsymbol{F}(\boldsymbol{y} + \varepsilon \boldsymbol{q}_k) - \boldsymbol{F}(\boldsymbol{y} - \varepsilon \boldsymbol{q}_k)}{2\varepsilon} \tag{19.32}$$

のような近似を用いて実際の Jacobian の計算を回避する [30, 31]．なお，ε はマシンイプシロン $\varepsilon_{machine}$ に対して

$$\varepsilon = \frac{\sqrt{\varepsilon_{machine}}}{\|\boldsymbol{q}_k\|_2} \tag{19.33}$$

のように選択する [30, 31]．

GMRES 法にはアルゴリズムで使用する部分空間の次元数を示すパラメータ m が存在するが[*9]，このパラメータ m を大きくすると大抵の場合 1 回 GMRES 法を適用することにより得られる解の精度が上がるものの，計算時間が増加する．そのため，複数回 GMRES 法を適用することにより所望の精度の結果を得る場合，m が小さすぎると GMRES 法の適用回数を増やすことになり，大きすぎると GMRES 法 1 回の計算時間を増やすことになる．また，m を大きくすることでいつでも GMRES 法の収束が速くなるとは限らないことが示されている [31]．そこで，Newton 法におけ

[*9] 詳細な理論については [4, Section 11.4.3] を参照．

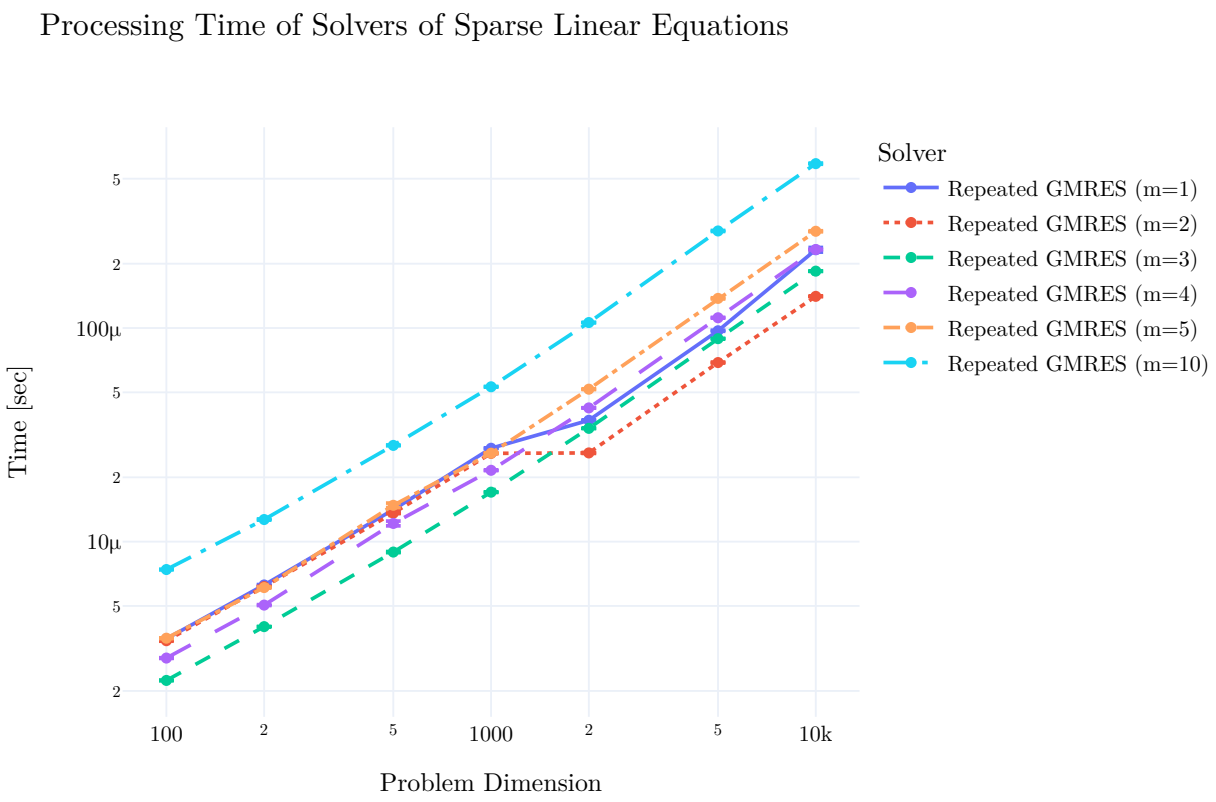


図 19.1: 繰り返し GMRES（Algorithm 19.4）を適用して線形方程式を解くのにかかった時間

る係数行列 $I - haJ$ をイメージした係数行列

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \alpha & 1 & \alpha & 0 & \cdots & 0 & 0 & 0 \\ 0 & \alpha & 1 & \alpha & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha & 1 & \alpha \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}, \qquad \alpha = 0.01 \tag{19.34}$$

を持つ方程式に対して GMRES 法を $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2 < 10^{-8}$ となるまで適用し，計算時間を計測した．その結果が図 19.1 である[*10]．この例では，部分空間の次元数 m を 2 か 3 にした場合に最も良い性能が得られている．なお，最初は α を小さくない 1 にしていたが，その場合は解の収束が極めて遅かった[*11]．GMRES 法を適用するにあたってはステップサイズが大きくなりすぎて係数行列 $I - haJ$ の非対角項が大きくならないように注意する必要がある．

文献 [30, 31] では GMRES 法を用いているが，係数行列を列ベクトルに左から作用させる以外の使い方をしないアルゴリズムであれば GMRES の代わりに使用できる．例えば，BiCGstab (Algorithm 4.5) を使用することがで

[*10] 計算に使用したマシンの CPU は Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz であり，コンパイラは Clang 14.0.6 を用いた．ソースコードは，リポジトリ [1] のコミット `e7cddd677dbd031f2b2c2e8e8622a8c318f7b31b` 時点におけるファイル `test/bench/ode/sparse_linear_equation_test.cpp` を使用している．

[*11] 1000 回反復しても残差 $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2 / \|\boldsymbol{b}\|_2$ の割合のオーダーが 0.1 程度になるという状況で，収束するまでの計算時間の計測を行えていない．

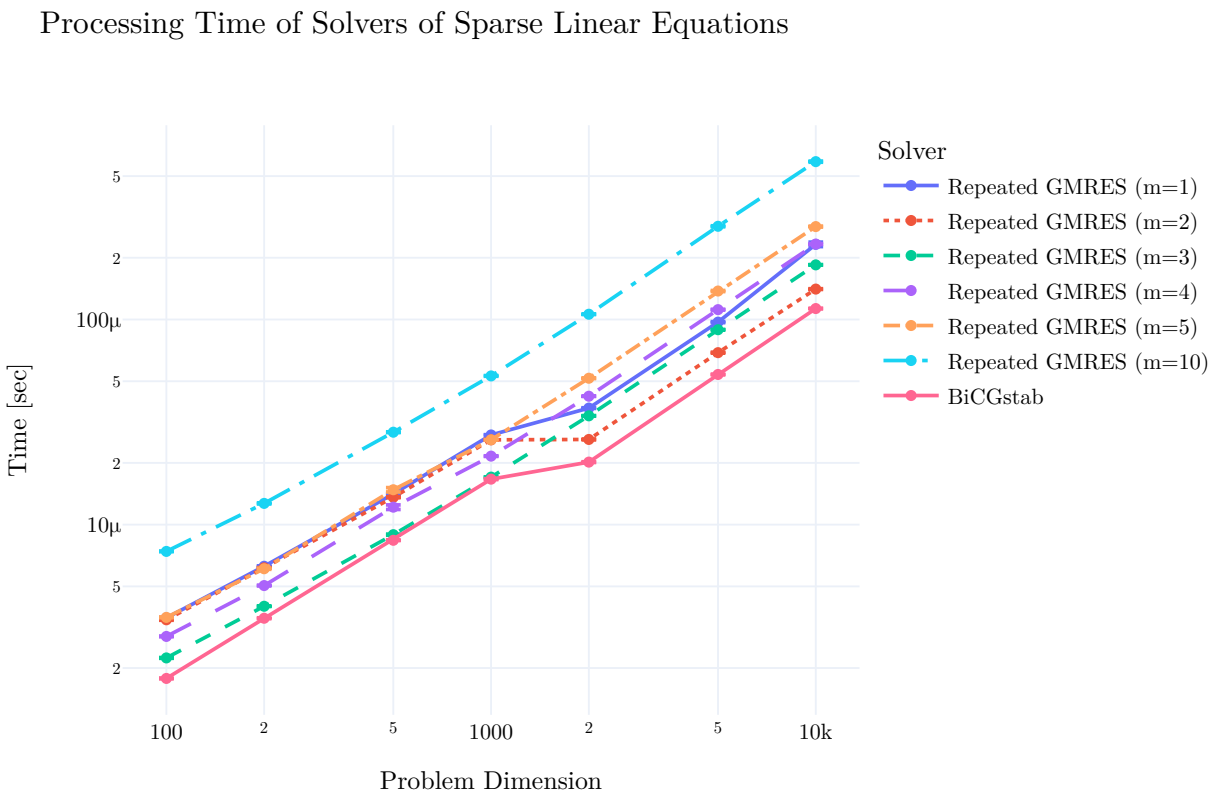


図 19.2: BiCGstab（Algorithm 4.5）を使用して線形方程式を解くのにかかった時間

きる．BiCGstab を使用して同様に式 (19.34) の係数行列を持つ方程式を解く時間を計測した結果が図 19.2 である．GMRES で最も速かった $m = 2, 3$ の場合よりもさらに速くなっている．

19.4 陽的公式の例

19.4.1 古典的 Runge-Kutta 法

古典的 Runge-Kutta 法と呼ばれる公式では，表 19.2 のような係数を用いる [26, 3.3 節]．単純な係数で次数 4 を達成できる．

19.4.2 RKF45 公式

RKF45 公式（RKF は Runge-Kutta-Fehlberg のこと）では，表 19.3 のような係数を用いる [26, 4.1 節 (a)], [32, Section 9.5]．この埋め込み型公式では，$\boldsymbol{k}_i$ から $\boldsymbol{y}(t+h)$ を算出する係数に 5 次の精度を持つ組と 4 次の精度を持つ組が存在する[*12]．

19.4.3 Euler 法

常微分方程式の解法としては最も基本的な Euler 法は形式的に Runge-Kutta 法とみなすことができる．Euler 法は

$$\boldsymbol{y}(t+h) \approx \boldsymbol{y}(t) + h\boldsymbol{f}(t, \boldsymbol{y}(t)) \tag{19.35}$$

表 19.2: 古典的 Runge-Kutta 法（RK4 公式）の Butcher 配列

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

表 19.3: RKF45 公式の Butcher 配列

0							
$\frac{1}{4}$	$\frac{1}{4}$						
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$					
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$				
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$			
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$		
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$	（5 次）
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0	（4 次）

表 19.4: Euler 法の Butcher 配列

1	
	1

[*12] 挙げた 2 件の文献のうち，文献 [26] では係数が一カ所誤っていたため注意が必要．

のように書かれるが，その Butcher 配列は表 19.4 に示す通りである．

19.5 半陰的公式の例

19.5.1 田中 Formula

文献 [25] では，田中正次氏が考案した半陰的・陰的公式がいくつか示されている．そのうち埋め込み型の半陰的公式を表 19.5, 19.6 に示す[*13]．

段数の多い陽的公式と同程度の精度を少ない段数で出すことができていることを確認できる．また，埋め込み型の 2 つの係数のうち次数の低い方の係数が単純になっている[*14]．

19.5.2 SDIRK

表 19.7 のように半陰的公式で a_{ii} が全て等しい値になっている場合は Singly Diagonally Implicit Runge-Kutta (SDIRK) 法と呼ばれる．SDIRK では，各段の計算を 19.3.1 節のように Newton-Raphson 法で行う際に解く方程式の係数行列

$$I - ha_{ii} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{t=t+b_i h, \boldsymbol{y}=\boldsymbol{y}(t)} \tag{19.36}$$

が 1 ステップの中では共有できるようになり，計算量の多い LU 分解の回数を減らすことができる．なお，表 19.7 の公式は stiffly accurate（19.3.3 節）である．

19.5.3 ESDIRK

SDIRK のように a_{ii} が $i = 2, 3, \ldots, s$ では等しい値になっている上に，$a_{11} = 0$ となるような公式は Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) と呼ばれる [33].

表 19.8 に [34] で示されている ESDIRK の公式の例を示す[*15]．

表 19.5: 田中 Formula1 公式の Butcher 配列

$\frac{13}{20}$	$\frac{13}{20}$			
$-\frac{1}{18}$	$-\frac{127}{180}$	$\frac{13}{20}$		
	$\frac{100}{127}$	$\frac{27}{127}$	（3 次）	
	1		（1 次）	

表 19.6: 田中 Formula2 公式の Butcher 配列

$\frac{133}{100}$	$\frac{133}{100}$			
$\frac{1}{2}$	$-\frac{5400}{18167}$	$\frac{28967}{36334}$		
$-\frac{33}{100}$	$\frac{133}{50}$	$-\frac{108}{25}$	$\frac{133}{100}$	
	$\frac{1250}{20667}$	$\frac{18167}{20667}$	$\frac{1250}{20667}$	（4 次）
	0	1		（2 次）

[*13] 文献 [25] には完全に陰的な 4 段 7 次の公式もあるが，係数がかなり複雑なため省略した．

[*14] 利用する際には高い次数の係数との差を見るように実装するため，次数の低い方の係数だけ単純でも計算コストへの影響はない．

[*15] [34] では同じ次数の陽的公式とセットで 3 次，4 次，5 次の公式が示されているが，4 次の公式が最も効率的だという実験結果が出ている．

19.6 陰的公式の例

19.6.1 Butcher-Kuntzmann 公式

文献 [26, 5.2 節 (b)] によると，陰的 Runge-Kutta 法では，s 段公式で $2s$ 次を超えることができないと証明されており，その限界の $2s$ 次の公式が存在することも証明されている．特に Legendre 関数の零点を用いて係数を決める種類の公式は，s 段 Butcher-Kuntzmann 公式と呼ばれる．2 段 Butcher-Kuntzmann 公式を表 19.9 に示す（前述の通り 4 次の精度を持つ）．

表 19.7: 4 次の SDIRK の例 [27, Section IV.6.]

$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{4}$				
$\frac{11}{20}$	$\frac{17}{50}$	$-\frac{1}{25}$	$\frac{1}{4}$			
$\frac{1}{2}$	$\frac{371}{1360}$	$-\frac{137}{2720}$	$\frac{15}{544}$	$\frac{1}{4}$		
1	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$-\frac{85}{12}$	$\frac{1}{4}$	
	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$-\frac{85}{12}$	$\frac{1}{4}$	（4 次）
	$\frac{59}{48}$	$-\frac{17}{96}$	$\frac{225}{32}$	$-\frac{85}{12}$	0	（3 次）

表 19.8: 4 次の ESDIRK の例（ARK4(3)6L[2]SA-ESDIRK [34]）

0							
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{83}{250}$	$\frac{8611}{62500}$	$-\frac{1743}{31250}$	$\frac{1}{4}$				
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$-\frac{654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$			
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$-\frac{71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$		
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$	
	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$	（4 次）
	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$	（3 次）

表 19.9: 2 段 Butcher-Kuntzmann 公式の Butcher 配列

$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$
$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

表 19.10: 陰的 Euler 法の Butcher 配列

1	1
	1

19.6.2 陰的 Euler 法

常微分方程式の解法として，（陽的）Euler 法と同様に基本的な陰的 Euler 法も形式的に Runge-Kutta 法とみなすことができる．陰的 Euler 法は

$$\boldsymbol{y}(t+h) \approx \boldsymbol{y}(t) + h\boldsymbol{f}(t, \boldsymbol{y}(t+h)) \tag{19.37}$$

のように書かれるが，その Butcher 配列は表 19.10 に示す通りである．この 1 段 2 次公式は陰的公式とも半陰的公式ともとることができる．

19.7 Rosenbrock 法

陰的 Runge-Kutta 法においては，方程式を Newton-Raphson 法などで反復的に解く必要がある．これにより，最終的な解の誤差には Runge-Kutta 法の離散化誤差に加えて Newton-Raphson 法の反復の打ち切りによる誤差も含まれるという問題がある．この問題を解決するための手法として，文献 [35] において Rosenbrock 法が考案された．

半陰的公式では

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h\sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j\right) \tag{19.38}$$

のような更新式を用いるが，右辺を $\boldsymbol{k}_i$ について 1 次近似すると

$$\boldsymbol{k}_i \approx \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j\right) + h a_{ii}\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}(t)+h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j} \boldsymbol{k}_i \tag{19.39}$$

のように書ける [35]．さらに，Jacobian と $\boldsymbol{k}_i$ の係数行列の LU 分解の計算回数を減らすため，Jacobian を固定し，$a_i i$ を i に依らない値にすることも考えられており，文献 [36] では次の形式の計算式を用いる[*16]．

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t + b_i h, \boldsymbol{y}(t) + h\sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j\right) + h\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}(t)}\left(\sum_{j=i}^{i-1}\gamma_{ij}\boldsymbol{k}_j + \gamma\boldsymbol{k}_i\right) \qquad \text{for } i = 1, 2, \ldots, s \tag{19.40}$$

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + \sum_{i=1}^{s} c_i \boldsymbol{k}_i \tag{19.41}$$

$\boldsymbol{k}_i$ の係数行列は i に依らず，$\boldsymbol{y}_(t)$ とステップ幅のみに依存するため，1 ステップごとに一度だけ Jacobian を計算し，$\boldsymbol{k}_i$ の係数行列の LU 分解を求めれば良い．さらに，陰的 Runge-Kutta 法と異なり，方程式は $\boldsymbol{k}_i$ について線形になっているため，Newton-Raphson 法の反復を行う必要はない．

半陰的 Runge-Kutta 法では各ステップの各 $i = 1, 2, \ldots, s$ ごとに LU 分解をやり直して Newton-Raphson 法の反復を行う必要があったが，Rosenbrock 法では各ステップの最初に LU 分解を行ったあと，各 $i = 1, 2, \ldots, s$ ごとに一度だけ LU 分解の結果をもとにした線形方程式の解を求めれば良い．対象となる常微分方程式の自由度を d としたとき，どちらの手法も計算時間のオーダーは d^3 となるが，その係数は Rosenbrock 法の方が小さくなる．

文献 [36] の ROS3w 公式では，次のような係数を用いる．

$$a_{21} = 6.666666666666666 \times 10^{-1} \tag{19.42}$$

$$a_{31} = 6.666666666666666 \times 10^{-1} \tag{19.43}$$

$$a_{32} = 0 \tag{19.44}$$

[*16] 文献 [36] 中の数式では時間微分の項も存在するが，提案されていた公式において時間微分の項は無視されていたため，省略する．

$$\gamma = 4.358665215084590 \times 10^{-1} \tag{19.45}$$
$$\gamma_{21} = 3.635068368900681 \times 10^{-1} \tag{19.46}$$
$$\gamma_{31} = -\ 8.996866791992636 \times 10^{-1} \tag{19.47}$$
$$\gamma_{32} = -\ 1.537997822626885 \times 10^{-1} \tag{19.48}$$
$$c_1 = 2.5 \times 10^{-1} \tag{19.49}$$
$$c_2 = 2.5 \times 10^{-1} \tag{19.50}$$
$$c_3 = 5 \times 10^{-1} \tag{19.51}$$
$$c_1^* = 7.467047032740110 \times 10^{-1} \tag{19.52}$$
$$c_2^* = 1.144064078371002 \times 10^{-1} \tag{19.53}$$
$$c_3^* = 1.388888888888889 \times 10^{-1} \tag{19.54}$$
$$b_i = \sum_{j=1}^{i-1} a_{ij} \tag{19.55}$$

c_i の係数は 3 次の精度を持ち，埋め込みの c_i^* の係数は 2 次の精度を持つ．さらに，4 次の精度を持つ埋め込み型公式も提案されている [37, 38] [*17][*18].

[*17] 文献 [37] は RODASP という公式を提案者本人が実装したものであり，提案された元の文献ではない．しかし，元の文献を入手できないためこの実装例を参考文献に挙げている．

[*18] これらの公式は [1] で実装した公式の中でも特に性能が良かったが，係数が多いため省略する．

第 20 章

平均ベクトル場法

エネルギー保存則を満たす系の運動方程式を解くことを目的とした手法の 1 つとして，平均ベクトル場法（Average vector field method, AVF method）[39] がある．

A 章で示しているように，座標 $\boldsymbol{q} \in \mathbb{R}^d$ とモーメント $\boldsymbol{p} \in \mathbb{R}^d$ によるハミルトニアン $H(\boldsymbol{q}, \boldsymbol{p})$ を用いた次の正準方程式を考える．

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}} \tag{20.1}$$

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}} \tag{20.2}$$

この式は変数 $\boldsymbol{y} = (\boldsymbol{q}^\top, \boldsymbol{p}^\top)^\top$ を用いて次のように書くこともできる．

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}) \equiv S\nabla H(\boldsymbol{y}) \tag{20.3}$$

文献 [39] はこの形式の微分方程式を前提として次のような数値解法を提案している．

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} = \int_0^1 \boldsymbol{f}((1-\xi)\boldsymbol{y}_n + \xi \boldsymbol{y}_{n+1}) d\xi \tag{20.4}$$

この手法がエネルギーを保存することを示す．まず，$\boldsymbol{f}(\boldsymbol{y}) \equiv S\nabla H(\boldsymbol{y})$ より

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} = S\int_0^1 \nabla H((1-\xi)\boldsymbol{y}_n + \xi \boldsymbol{y}_{n+1}) d\xi \tag{20.5}$$

である．ここで，S は

$$S = \begin{pmatrix} O & I \\ -I & O \end{pmatrix} \tag{20.6}$$

であるため[*1]，任意のベクトル $\boldsymbol{y} = (\boldsymbol{q}^\top, \boldsymbol{p}^\top)^\top$ において，

$$\boldsymbol{y}^\top S \boldsymbol{y} = \begin{pmatrix} \boldsymbol{q}^\top & \boldsymbol{p}^\top \end{pmatrix} \begin{pmatrix} O & I \\ -I & O \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}^\top & \boldsymbol{p}^\top \end{pmatrix} \begin{pmatrix} \boldsymbol{p} \\ -\boldsymbol{q} \end{pmatrix} = \boldsymbol{0} \tag{20.7}$$

となる．よって，式 (20.5) の両辺と $\int_0^1 \nabla H((1-\xi)\boldsymbol{y}_n + \xi \boldsymbol{y}_{n+1}) d\xi$ の内積をとると

$$\frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} \int_0^1 \nabla H((1-\xi)\boldsymbol{y}_n + \xi \boldsymbol{y}_{n+1}) d\xi = 0 \tag{20.8}$$

[*1] このような行列は歪対称行列と呼ばれる．

となる．さらに左辺は

$$\frac{\boldsymbol{y}_{n+1}-\boldsymbol{y}_n}{h}\int_0^1 \nabla H((1-\xi)\boldsymbol{y}_n+\xi\boldsymbol{y}_{n+1})d\xi \tag{20.9}$$

$$=\frac{1}{h}\int_0^1 \frac{d}{d\xi}H((1-\xi)\boldsymbol{y}_n+\xi\boldsymbol{y}_{n+1})d\xi \tag{20.10}$$

$$=\frac{H(\boldsymbol{y}_{n+1})-H()\boldsymbol{y}_n)}{h} \tag{20.11}$$

と変形できるから，次のようにエネルギーの保存が示される．

$$H(\boldsymbol{y}_{n+1})-H(\boldsymbol{y}_n)=0 \tag{20.12}$$

式 (20.4) は 2 次の精度だが，3 次と 4 次の公式も文献 [39] で示されている．それらは次の式で与えられる．

$$\frac{\boldsymbol{y}_{n+1}-\boldsymbol{y}_n}{h}=\left(I+\alpha h^2\left(\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\hat{\boldsymbol{y}}}\right)^2\right)\int_0^1 \boldsymbol{f}((1-\xi)\boldsymbol{y}_n+\xi\boldsymbol{y}_{n+1})d\xi \tag{20.13}$$

パラメータと次数は次のようになる．

- $\alpha=0$ とすると 2 次精度の更新式 (20.4) が得られる．
- $\alpha=-1/12$, $\hat{\boldsymbol{y}}=\boldsymbol{y}_n$ とすると 3 次精度の更新式が得られる．
- $\alpha=-1/12$, $\hat{\boldsymbol{y}}=(\boldsymbol{y}_n+\boldsymbol{y}_{n+1})/2$ とすると 4 次精度の更新式が得られる．

この手法の更新式は次数に依らず陰的で，次数が上がるごとに計算が複雑になっていく．ここで，次数 2 のときの方程式

$$F(\boldsymbol{y}_{n+1})\equiv\frac{\boldsymbol{y}_{n+1}-\boldsymbol{y}_n}{h}-\int_0^1 \boldsymbol{f}((1-\xi)\boldsymbol{y}_n+\xi\boldsymbol{y}_{n+1})d\xi=0 \tag{20.14}$$

を考える．これを Newton-Raphson 法（17 章）で解くには，Jacobian が必要となる．Jacobian は次のようになる．

$$\frac{\partial F}{\partial \boldsymbol{y}_{n+1}}=\frac{1}{h}I-\int_0^1 \xi\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=(1-\xi)\boldsymbol{y}_n+\xi\boldsymbol{y}_{n+1}} d\xi \tag{20.15}$$

さらに，Jacobian の変化が十分小さいとすると，

$$\frac{\partial F}{\partial \boldsymbol{y}_{n+1}} \tag{20.16}$$

$$\approx\frac{1}{h}I-\int_0^1 \xi\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}_n} d\xi \tag{20.17}$$

$$=\frac{1}{h}I-\frac{1}{2}\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}_n} \tag{20.18}$$

のように近似できる．3 次と 4 次の更新式でも，Newton-Raphson 法の Jacobian においては h^2 の項を無視して式 (20.18) で近似することができる．

第 21 章

シンプレクティック積分法

平均ベクトル場法（20 章）と同様に，位置 $\boldsymbol{q}$，モーメント $\boldsymbol{p}$ に対するハミルトニアン $H(\boldsymbol{q}, \boldsymbol{p})$ から得られる正準方程式

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}} \tag{21.1}$$

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}} \tag{21.2}$$

を解くための手法の 1 つにシンプレクティック積分法 (symplectic integrator) と呼ばれる手法がある [28, Section II.16.]．シンプレクティック積分法は，時間変化に伴う点 $(\boldsymbol{q}, \boldsymbol{p})$ の変化により

$$\omega^2 = \sum_{i=1}^{n} dp_i \wedge dq_i \tag{21.3}$$

という量が変化しないという特性を持つ．そのため，周期運動をする自励系において長時間に渡るシミュレーションを行ってもステップ幅を固定していればエネルギーの誤差が増大しないという利点がある [40] [*1]．

ここでは，シンプレクティックな Runge-Kutta 法を示す．

21.1 陰的 Runge-Kutta 法

陰的 Runge-Kutta 法の中にはシンプレクティックとなっているものが存在する．例えば，表 19.9 のような s 段 Butcher-Kuntzmann 公式はシンプレクティックである [28, Section II.16.].

21.2 可分ハミルトニアンに対する陽的な分離型 Runge-Kutta 法

通常の陽的 Runge-Kutta 法はシンプレクティックになり得ないことが証明されているが，ハミルトニアンが可分（$H(\boldsymbol{q}) = T(\boldsymbol{p}) + V(\boldsymbol{q})$ のように $\boldsymbol{q}$ の項と $\boldsymbol{p}$ の項に分割可能）な場合，$\boldsymbol{q}$ と $\boldsymbol{p}$ を交互に更新する以下のような分離型 Runge-Kutta 法を考えることにより，陽的でシンプレクティックな積分公式を得ることができる [28, Section II.16.].

$$\boldsymbol{P}_i = \boldsymbol{p}(t) + h \sum_j a_{ij} \boldsymbol{k}_j, \qquad \boldsymbol{Q}_i = \boldsymbol{q}(t) + h \sum_j \hat{a}_{ij} \boldsymbol{l}_j \tag{21.4}$$

$$\boldsymbol{p}(t+h) = \boldsymbol{p}(t) + h \sum_j b_j \boldsymbol{k}_j, \qquad \boldsymbol{q}(t+h) = \boldsymbol{q}(t) + h \sum_j \hat{b}_j \boldsymbol{l}_j, \tag{21.5}$$

[*1] ステップ幅を固定しない場合はエネルギーが増大し，解が不安定になることが実験で確認されている [28, Section II.16.].

Algorithm 21.1 可分ハミルトニアンに対する陽的な分離型 Runge-Kutta 法 [28, Section II.16.]

procedure SynplecticExplicitRungeKutta($\frac{\partial V}{\partial \boldsymbol{q}}, \frac{\partial T}{\partial \boldsymbol{p}}, t, \boldsymbol{q}(t), \boldsymbol{p}(t), h$)
  $\boldsymbol{Q} \leftarrow \boldsymbol{q}(t)$ ▷ $\boldsymbol{Q}_1$
  $\boldsymbol{P} \leftarrow \boldsymbol{p}(t)$ ▷ $\boldsymbol{P}_0$
  for $i \leftarrow 1, s$ **do** ▷ s は段数
    $\boldsymbol{P} \leftarrow \boldsymbol{P} - hb_i \frac{\partial V}{\partial \boldsymbol{q}}(\boldsymbol{Q})$ ▷ $\boldsymbol{P}_{i-1}$ と $\boldsymbol{Q}_i$ から $\boldsymbol{P}_i$ を算出
    $\boldsymbol{Q} \leftarrow \boldsymbol{Q} + h\hat{b}_i \frac{\partial T}{\partial \boldsymbol{p}}(\boldsymbol{P})$ ▷ $\boldsymbol{P}_i$ と $\boldsymbol{Q}_i$ から $\boldsymbol{Q}_{i+1}$ を算出
  end for
  $q(t+h) \leftarrow \boldsymbol{Q}$
  $p(t+h) \leftarrow \boldsymbol{P}$
  return $(q(t+h), p(t+h))$
end procedure

$$\boldsymbol{k}_i = -\frac{\partial H}{\partial \boldsymbol{q}}(\boldsymbol{P}_i, \boldsymbol{Q}_i) = -\frac{\partial V}{\partial \boldsymbol{q}}(\boldsymbol{Q}_i), \qquad \boldsymbol{l}_i = \frac{\partial H}{\partial \boldsymbol{p}}(\boldsymbol{P}_i, \boldsymbol{Q}_i) = \frac{\partial T}{\partial \boldsymbol{p}}(\boldsymbol{P}_i) \tag{21.6}$$

ここで，$a_{ij} = 0$ $(i < j)$, $\hat{a}_{ij} = 0$ $(i \leq j)$ とすると，シンプレクティックであるための条件は $a_{ij} = b_j$ $(i \leq j)$, $\hat{a}_{ij} = \hat{b}_j$ $(i > j)$ であることが示されており [28, Section II.16.]，$\boldsymbol{P}_i$ と $\boldsymbol{Q}_i$ は次のようになる．

$$\boldsymbol{P}_i = \boldsymbol{P}_{i-1} + hb_i \boldsymbol{k}_j = \boldsymbol{P}_{i-1} - hb_i \frac{\partial V}{\partial \boldsymbol{q}}(\boldsymbol{Q}_i), \qquad \boldsymbol{P}_0 = \boldsymbol{p}(t) \tag{21.7}$$

$$\boldsymbol{Q}_i = \boldsymbol{Q}_{i-1} + h\hat{b}_{i-1} \boldsymbol{l}_{i-1} = \boldsymbol{Q}_{i-1} + h\hat{b}_{i-1} \frac{\partial T}{\partial \boldsymbol{p}}(\boldsymbol{P}_{i-1}), \qquad \boldsymbol{Q}_1 = \boldsymbol{q}(t) \tag{21.8}$$

このとき，$\boldsymbol{P}_i$ と $\boldsymbol{Q}_i$ は以下のように交互に計算できる．

1. $\boldsymbol{P}_0$ と $\boldsymbol{Q}_1$ から $\boldsymbol{P}_1$ を算出する．
2. $\boldsymbol{P}_1$ と $\boldsymbol{Q}_1$ から $\boldsymbol{Q}_2$ を算出する．
3. $\boldsymbol{P}_1$ と $\boldsymbol{Q}_2$ から $\boldsymbol{P}_2$ を算出する．
4. $\boldsymbol{P}_2$ と $\boldsymbol{Q}_2$ から $\boldsymbol{Q}_3$ を算出する．
5. 以降同様に続ける．

これらをまとめると，Algorithm 21.1 となる．

この手法における係数 b_i, $\hat{b}_i$ を表 21.1 のようにまとめたものが，表 21.2, 21.3, 21.4, 21.5, 21.6 である．

ここで，4 次の公式のうち表 21.5 は表 21.4 の公式を 2 つ結合して得られたものであるのに対し，表 21.6 は代数的に 4 次の公式を算出したものとなっている．同様に代数的に得られた 4 段 4 次の公式が文献 [42] に示されている．

表 21.1: 陽的シンプレクティック積分法の係数の表記

b	b_1	b_2	$\dots$	b_s
$\hat{b}$	$\hat{b}_1$	$\hat{b}_2$	$\dots$	$\hat{b}_s$

表 21.2: 1 段 1 次の陽的シンプレクティック積分法 [28, Section II.16.]

b	1
$\hat{b}$	1

表 21.3: 2 段 2 次の陽的シンプレクティック積分法（Leap-frog 法）[41]

b	$\frac{1}{2}$	$\frac{1}{2}$
$\hat{b}$	1	0

表 21.4: 3 段 3 次の陽的シンプレクティック積分法 [28, Section II.16.]

b	$\frac{7}{24}$	$\frac{3}{4}$	$-\frac{1}{24}$
$\hat{b}$	$\frac{2}{3}$	$-\frac{2}{3}$	-1

表 21.5: 6 段 4 次の陽的シンプレクティック積分法 [28, Section II.16.]

b	$\frac{7}{48}$	$\frac{3}{8}$	$-\frac{1}{48}$	$-\frac{1}{48}$	$\frac{3}{8}$	$\frac{7}{48}$
$\hat{b}$	$\frac{1}{3}$	$-\frac{1}{3}$	1	$-\frac{1}{3}$	$\frac{1}{3}$	0

表 21.6: 4 段 4 次の陽的シンプレクティック積分法（$\alpha = 1 - 2^{1/3}$）[41]

b	$\frac{1}{2(1+\alpha)}$	$\frac{\alpha}{2(1+\alpha)}$	$\frac{\alpha}{2(1+\alpha)}$	$\frac{1}{2(1+\alpha)}$
$\hat{b}$	$\frac{1}{1+\alpha}$	$\frac{\alpha-1}{1+\alpha}$	$\frac{1}{1+\alpha}$	0

第 22 章

数値実験

この部において説明してきた常微分方程式の数値解法について，数値実験を行った結果を示す．

22.1　対象とした数値解法

本章で対象とした数値解法について，結果を示す図の中で用いる略称とそれに対応する解法を以下に示す．

- Runge-Kutta 法（19 章）
  - 陽的公式

    RK4　古典的な 4 次の Runge-Kutta 法 [26]（表 19.2）

    RKF45　RKF45 公式 [26]（表 19.3）

    DOPRI5　DOPRI5 公式 [27]

    ARK4(3)-ERK　ARK4(3)6L[2]SA-ERK 公式 [34]
  - 半陰的公式

    Tanaka1　田中 Formula1 公式 [25]（表 19.5）

    Tanaka2　田中 Formula2 公式 [25]（表 19.6）

    SDIRK4　4 次の SDIRK [27, Section IV.6.]（表 19.7）

    ARK4(3)-ESDIRK　ARK4(3)6L[2]SA-ESDIRK 公式 [34]（表 19.8）

    ARK5(4)-ESDIRK　ARK5(4)6L[2]SA-ESDIRK 公式 [34]

    ESDIRK45c　ESDIRK45c 公式 [33]
- Rosenbrock 法（19.7 章）

  ROS3w　ROS3w 公式 [36]

  ROS34PW3　ROS34PW3 公式 [36]

  RODASP　RODASP 公式 [37]

  RODASPR　RODASPR 公式 [38]
- 平均ベクトル場法（20 章）

  AVF2　2 次精度の解法

  AVF3　3 次精度の解法

  AVF4　4 次精度の解法
- シンプレクティック積分法（21 章）

  LeapFrog　2 段 2 次の陽的シンプレクティック積分法（Leap-frog 法）[41]（表 21.3）

  Forest4　4 段 4 次の陽的シンプレクティック積分法 [41]（表 21.6）

22.2　振り子の運動

以下の式で表される簡易化した振り子の運動方程式に対して数値解法を適用した結果を示す[*1]．

$$\ddot{\theta} = -\sin\theta \tag{22.1}$$

振れ角が小さいものとして $\sin\theta \approx \theta$ と近似すれば解 $\theta(t)$ を三角関数で表せるが，ここでは近似しない方程式に対して数値解法を適用した．

まず，ステップ幅を自動決定しながら各種数値解法を適用した．ここで，ステップ幅は 19.1.2 節で示したアルゴリズムにより決定した．$t = 0$ における初期値 $\theta = 1$ をもとに $t = 10$ における θ の値を求めるという初期値問題で評価を行った結果を図 22.1, 22.2 に示す．振り子の運動方程式を解くにあたっては，単純な Runge-Kutta 法の陽的公式が最も速かった．その Runge-Kutta 法の陽的公式 3 種類ではおおよそ同等の性能となった．Runge-Kutta 法の半陰的公式の中では性能向上が行われている SDIRK, ESDIRK の公式による解法が比較的速かった．SDIRK からアルゴリズム自体も変更して性能向上が行われている Rosenbrock 法は SDIRK よりもさらに速かった．平均ベクトル場法は計算が少し複雑になるため，性能は比較的悪くなった．

続いて，ステップ幅を固定して数値解法を適用した．ここでは，主にステップ幅を固定して使用することを前提としているシンプレクティック積分法の評価を目的としている．$t = 0$ における初期値 $\theta = 1$ をもとに $t = 100$ における θ の値を求めるという初期値問題で評価を行った結果を図 22.3 に示す．シンプレクティック積分法 2 種類の中では 4 次の解法（Forest4）の方が速かった．

[*1] 厳密には質量や長さといったパラメータが係数として必要だが，ここではパラメータ値 1 として省略している．

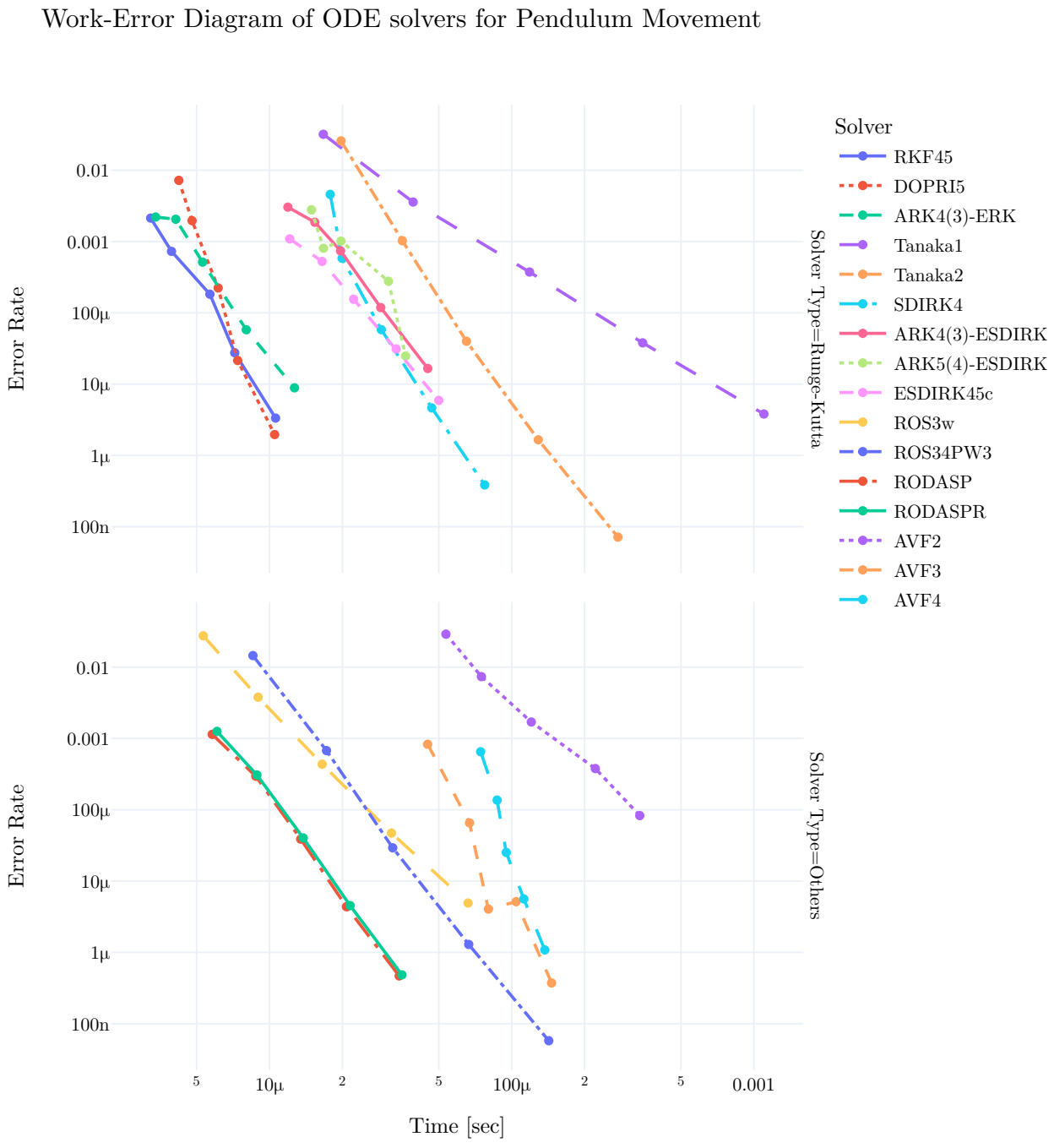


図 22.1: 振り子の運動方程式に数値解法を適用したときの計算時間と誤差（ステップ幅の自動調整機能 [29] 付きの解法によるもの）

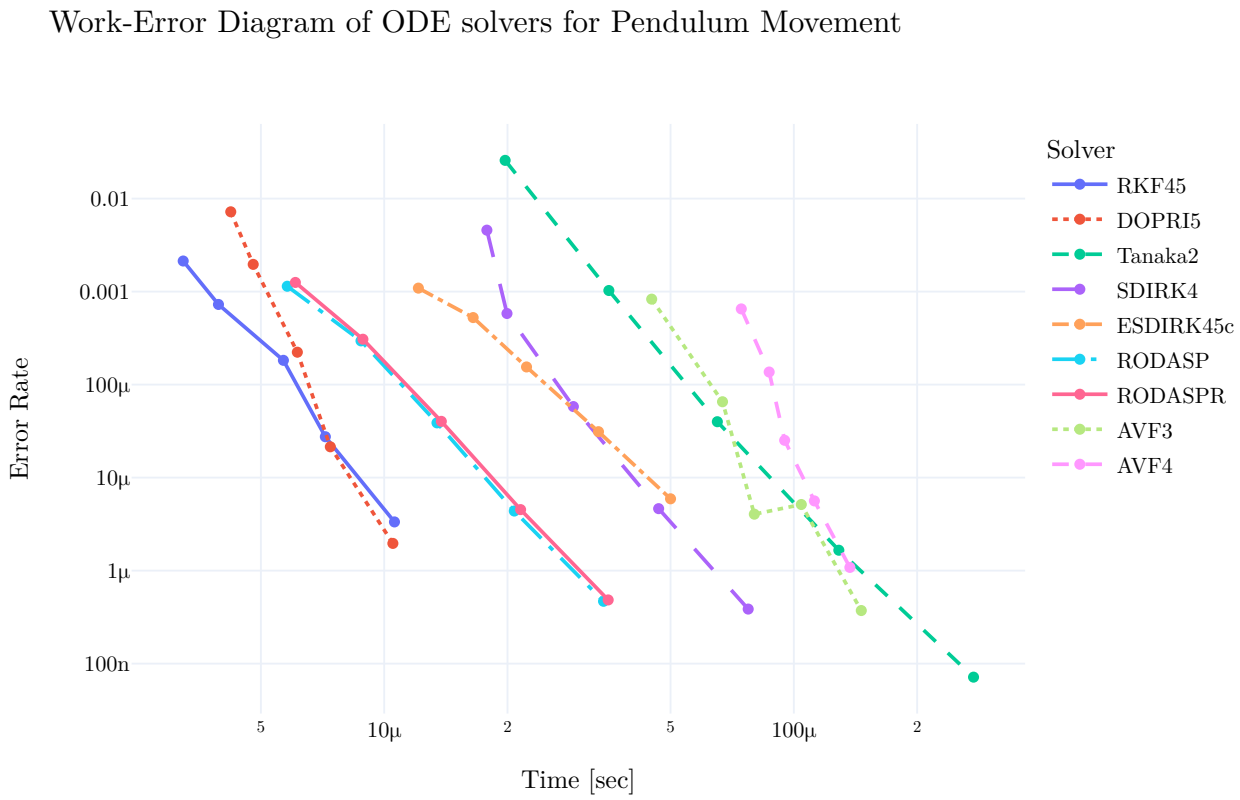


図 22.2: 図 22.1 から種類ごとに比較的速いアルゴリズムだけ抜き出したもの

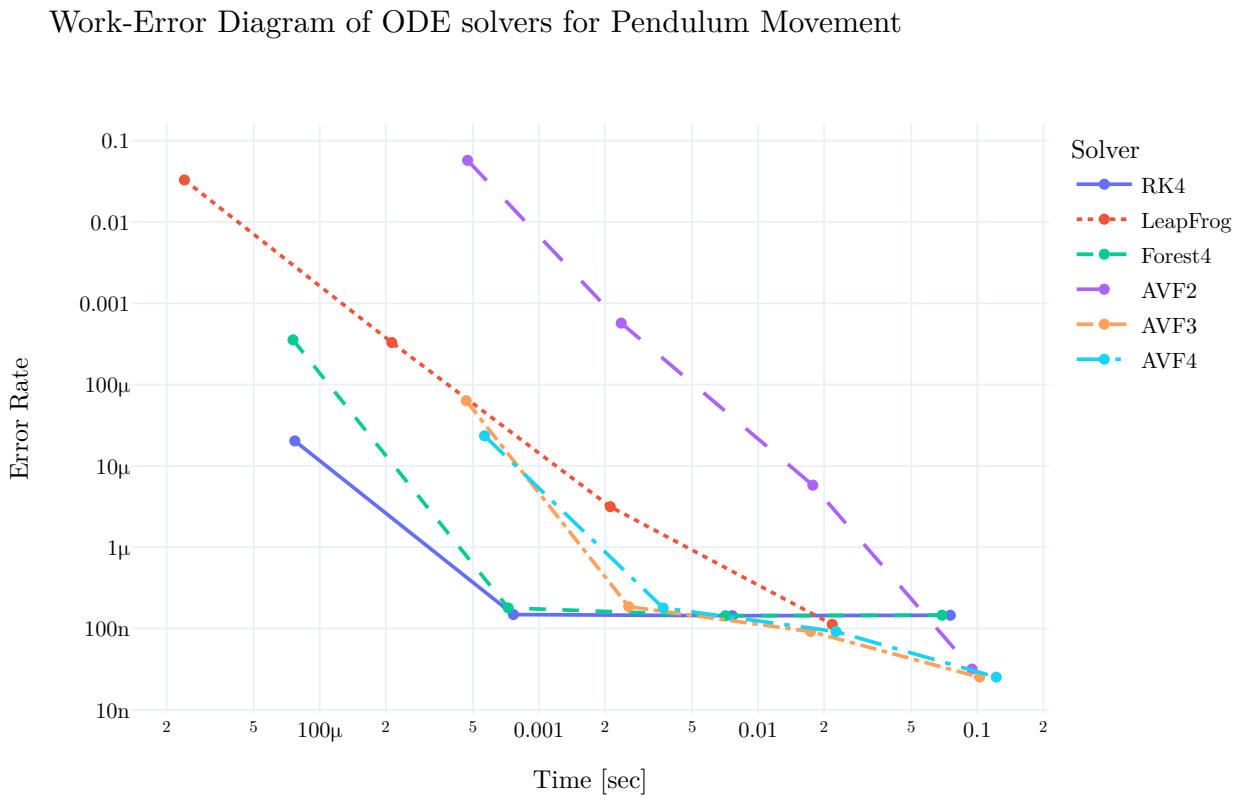


図 22.3: 振り子の運動方程式に数値解法を適用したときの計算時間と誤差（ステップ幅を固定した解法によるもの）

22.3　Kaps の問題（硬い系の例）

Kaps の問題 (Kaps' problem) は文献 [34] において常微分方程式の解法の比較に用いられており，以下の式で表される．

$$\dot{y_1} = -(\varepsilon^{-1} + 2)y_1 + \varepsilon^{-1}y_2^2 \tag{22.2}$$

$$\dot{y_2} = y_1 - y_2 - y_2^2 \tag{22.3}$$

この常微分方程式はパラメータ ε が小さいほど硬い系になるが，ε の値に関係なく以下の解をもつ．

$$y_1 = \exp(-2t) \tag{22.4}$$

$$y_2 = \exp(-t) \tag{22.5}$$

この常微分方程式に対して各種数値解法を適用した．ここで，ステップ幅は 19.1.2 節で示したアルゴリズムにより決定した．$t = 0$ における初期値をもとに $t = 1$ における数値解を求めるという初期値問題で評価を行った結果を図 22.4 に示す．$\varepsilon = 1$ の場合は陽的解法が最も速いが，ε が小さくなると陽的解法は計算に時間がかかるようになった．一方，陰的解法では特に計算時間の増加が見られなかった．この結果により，硬い系では陽的解法で安定した解を求めるために必要となるステップ幅が小さくなり，同じ時刻の数値解を求めるために必要な計算時間が長くなってしまうという事実を確認することができた．なお，ε の値と関係なく陰的解法の中では Rosenbrock 法の RODASP, RODASPR 公式が速かった．

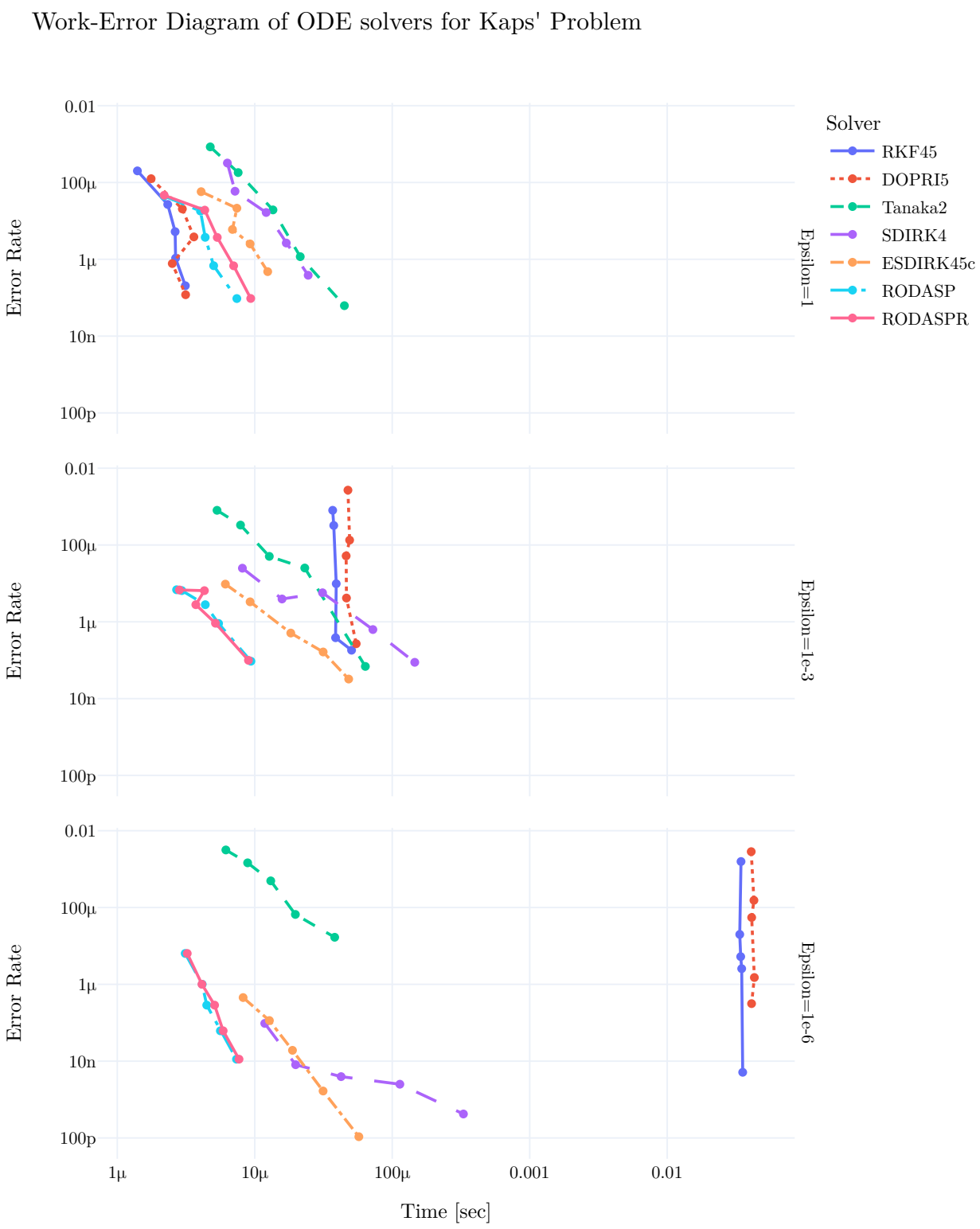


図 22.4: Kaps の問題にに数値解法を適用したときの計算時間と誤差（ステップ幅の自動調整機能 [29] 付きの解法によるもの）

第 V 部

高精度演算

第 23 章

加算による丸め誤差の低減

大きさのことなる数値を大量に加算していくにあたって，丸め誤差の影響で総和の誤差が大きくなる場合がある．例えば，

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \ldots = \frac{\pi^2}{6} \tag{23.1}$$

のような総和を前から順に加算していく場合，後に加算する数値は下の方の桁が総和へ加算されないという問題がある．この誤差は積み残しと呼ばれている [25, 4.4 節].

このような積み残しの問題を避ける簡単な方法は，できる限り小さい数値から順に加算していくようにすることである．しかし，

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \tag{23.2}$$

のように前から順に加算する数値が求まっていく場合に後ろの方の値が小さい項から加算していくのは効率が悪い．また，適応積分（8 章）のようにランダムに大きさの異なる数値を加算していく場合もある．そこで，加算の仕方を工夫することにより積み残しによる誤差を少なくする手法について示す．

23.1 Kahan Summation

Kahan Summation [43] は積み残しを保持するための変数を用意することで積み残しの誤差を低減する手法の 1 つである．Algorithm 23.1 のように計算を行う．

なお，実装時に一時変数のオーバーヘッドを減らしやすいように変数を入れ替えると，Algorithm 23.2 のようになる．計算される値自体は変わらない．

23.2 敗者復活算法

敗者復活算法 [25] も積み残しを保持するための変数を用意することで積み残しの誤差を低減する手法の 1 つである．Algorithm 23.3 のように計算を行う．Kahan Summation (Algorithm 23.1) と少し手順が異なるものの，計算される値は同じである．

Algorithm 23.1 Kahan Summation [43]

```
1: procedure PerformKahanSummation(y_1, y_2, ..., y_n)
2:     s ← 0                                  ▷ 総和を保持する変数
3:     r ← 0                                  ▷ 積み残しを保持する変数
4:     for i = 1, 2, ..., n do
5:         r ← r + y_i
6:         t ← s + r                          ▷ 一旦加算してみる
7:         r ← r − (t − s)                    ▷ 積み残しを更新
8:         s ← t                              ▷ 総和を更新
9:     end for
10:    return s
11: end procedure
```

Algorithm 23.2 Algorithm 23.1 を一時変数が少なくなるよう変更したもの

```
1: procedure PerformKahanSummation(y_1, y_2, ..., y_n)
2:     s ← 0                                  ▷ 総和を保持する変数
3:     r ← 0                                  ▷ 積み残しを保持する変数
4:     for i = 1, 2, ..., n do
5:         p ← s                              ▷ 加算前の総和を保持する変数
6:         r ← r + y_i
7:         s ← s + r                          ▷ 総和を更新
8:         r ← r − (s − p)                    ▷ 積み残しを更新
9:     end for
10:    return s
11: end procedure
```

Algorithm 23.3 敗者復活算法 [25]

```
1: procedure PerformRepechangeArithmetic(y_1, y_2, ..., y_n)
2:     s ← 0                                  ▷ 総和を保持する変数
3:     r ← 0                                  ▷ 積み残しを保持する変数
4:     for i = 1, 2, ..., n do
5:         t ← s + r + y_i                    ▷ 一旦加算してみる
6:         d ← t − s
7:         r ← r + y_i − d                    ▷ 積み残しを更新
8:         s ← t                              ▷ 総和を更新
9:     end for
10:    return s
11: end procedure
```

第 24 章

倍精度浮動小数点数による多倍長浮動小数点数

倍精度浮動小数点数を用いて四倍精度，八倍精度といったより高い精度の演算を行う手法が考えられている [44, 45]．それぞれ 2 つ，4 つの倍精度浮動小数点数の和で小数を表現し，仮数の桁が被らないように演算することで高い精度を実現する．

24.1 記号

本章で使用する記号を以下に示す．

- $\oplus$ 倍精度浮動小数点数の加算
- $\ominus$ 倍精度浮動小数点数の減算
- $\otimes$ 倍精度浮動小数点数の乗算
- $\oslash$ 倍精度浮動小数点数の除算

24.2 基本演算

倍精度浮動小数点数による四倍精度，八倍精度演算の既存文献 [45] で使用される基本的な演算を以下にまとめる．

まず，$|a| \geq |b|$ となる倍精度浮動小数点数 a, b について $s = a \oplus b$, $a + b = s + e$ が成り立つ倍精度浮動小数点数 s, e を算出するアルゴリズムを Algorithm 24.1 に示す．なお，a, b の大小が不明な場合は Algorithm 24.2 を使用する．

また，乗算についても $s = a \otimes b$, $a \times b = s + e$ となる倍精度浮動小数点数 s, e を算出する Algorithm 24.3 が存在する．ただし，fused multiply-add (FMA) 命令が存在する CPU では，Algorithm 24.4 により高速化が期待できる．

Algorithm 24.1 大小の明確な倍精度浮動小数点数の加算と誤差計算 [45, Algorithm 3]

procedure QuickTwoSum(a, b)
    $s \leftarrow a \oplus b$
    $e \leftarrow b \ominus (s \ominus a)$
    return (s, e)
end procedure

Algorithm 24.2 大小の不明な倍精度浮動小数点数の加算と誤差計算 [45, Algorithm 4]

procedure TwoSum(a, b)
    $s \leftarrow a \oplus b$
    $v \leftarrow s \ominus a$
    $e \leftarrow (a \ominus (s \ominus v)) \oplus (b \ominus v)$
    return (s, e)
end procedure

Algorithm 24.3 倍精度浮動小数点数の乗算と誤差計算 [45, Algorithm 5, 6]

procedure TwoProd(a, b)
    $p \leftarrow a \otimes b$
    $(a_h, a_l) \leftarrow \mathrm{Split}(a)$
    $(b_h, b_l) \leftarrow \mathrm{Split}(b)$
    $e \leftarrow ((a_h \otimes b_h \ominus p) \oplus a_h \otimes b_l \oplus a_l \otimes b_h) \oplus a_l \otimes b_l$
    return (p, e)
end procedure

procedure Split(a)
    $t \leftarrow (2^{27} + 1) \otimes a$
    $a_h \leftarrow t \ominus (t \ominus a)$
    $a_l \leftarrow a \ominus a_h$
    return (a_h, a_l)
end procedure

Algorithm 24.4 倍精度浮動小数点数の乗算と誤差計算（FMA 命令を使用する場合）[45, Algorithm 7]

procedure TwoProdFMA(a, b)
    $s \leftarrow a \otimes b$
    $e \leftarrow \mathrm{FMA}(a, b, -p)$　　▷ $\mathrm{FMA}(a, b, c)$ は $a \times b + c$ を $a \times b$ を途中で丸めずに計算する．
    return (s, e)
end procedure

Algorithm 24.5 四倍精度の加算（正確な演算）[47]

procedure QuadAddPrecisely($(a_h, a_l), (b_h, b_l)$)
  $(x_h, x_l) \leftarrow \mathrm{TwoSum}(a_h, b_h)$
  $(y_h, y_l) \leftarrow \mathrm{TwoSum}(a_l, b_l)$
  $x_l \leftarrow x_l \oplus y_h$
  $(x_h, x_l) \leftarrow \mathrm{QuickTwoSum}(x_h, x_l)$
  $x_l \leftarrow x_l \oplus y_l$
  $(x_h, x_l) \leftarrow \mathrm{QuickTwoSum}(x_h, x_l)$
  return (x_h, x_l)
end procedure

Algorithm 24.6 四倍精度の加算（簡潔な演算）[44, 46]

procedure QuadAddSimple($(a_h, a_l), (b_h, b_l)$)
  $(x_h, x_l) \leftarrow \mathrm{TwoSum}(a_h, b_h)$
  $x_l \leftarrow x_l \oplus a_l \oplus b_l$
  $(x_h, x_l) \leftarrow \mathrm{QuickTwoSum}(x_h, x_l)$
  return (x_h, x_l)
end procedure

Algorithm 24.7 四倍精度の乗算 [46, 47]

procedure QuadMultiply($(a_h, a_l), (b_h, b_l)$)
  $(x_h, x_l) \leftarrow \mathrm{TwoProd}(a_h, b_h)$
  $x_l \leftarrow x_l \oplus a_h \otimes b_l \oplus a_l \otimes b_h$
  $(x_h, x_l) \leftarrow \mathrm{QuickTwoSum}(x_h, x_l)$
  return (x_h, x_l)
end procedure

24.3 倍精度浮動小数点数による四倍精度演算

四倍精度浮動小数点数を $a = a_h + a_l$, $|a_l| \leq (1/2)\,\mathrm{ulp}(|a_h|)$ となる倍精度浮動小数点数 a_h, a_l で表現する．

24.3.1 四則演算

このとき，加算は Algorithm 24.5, 24.6 のようなアルゴリズムで行うことができる．Algorithm 24.6 の方が簡潔な演算手法だが，倍精度浮動小数点数の 2 倍の 104 ビットの精度で演算できることが示されている [46]．減算も加算と同様にして行うことができる．

また，乗算は Algorithm 24.7 のようにすることで 102 ビットの精度で算出できることが示されている [46]．

残りの四則演算である除算は Algorithm 24.8 のようにすることで算出できる [48]．

Algorithm 24.8 四倍精度の除算 [48]

1: **procedure** QUADDIVIDE($(a_h, a_l), (b_h, b_l)$)
2: $c \leftarrow 1 \oslash b_h$
3: $d \leftarrow b_l \otimes c$
4: $x_h \leftarrow a_h \otimes c$
5: $(r_1, r_2) \leftarrow \mathrm{TwoProd}(x_h, b_h)$
6: $x_l \leftarrow ((a_h \ominus r_1) \ominus r_2) \otimes c$
7: $x_l \leftarrow x_l \oplus x_h \otimes ((a_l \oslash a_h) \ominus d)$
8: $(x_h, x_l) \leftarrow \mathrm{QuickTwoSum}(x_h, x_l)$
9: **return** (x_h, x_l)
10: **end procedure**

第 VI 部

微分

第 25 章

導入

最適化や方程式の数値解法など，関数の微分が必要になるアルゴリズムは多々ある．そこで，本部では関数の微分を行う方法をまとめる．

第 26 章

自動微分

自動微分 (automatic differentiation) では，関数 $\boldsymbol{f} : \mathbb{R}^m \to \mathbb{R}^n$ の関数値 $\boldsymbol{f}(\boldsymbol{x})$ を求めるために必要な演算をもとに Jacobian $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ を自動で算出する [49].

自動微分では，微分／偏微分における合成関数の微分公式

$$\frac{\partial \boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))}{\partial \boldsymbol{g}(\boldsymbol{x})} \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}} \tag{26.1}$$

を利用する．右辺値を右側から計算していくか，左側から計算していくかにより，それぞれ前進型自動微分 (forward-mode automatic differentiation)，後退型自動微分 (backward-mode automatic differentiation) と呼び分けられる．

26.1 前進型自動微分

前進型自動微分では，式 (26.1) における $\boldsymbol{g}(\boldsymbol{x})$ の Jacobian が分かっている状態から $\boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))$ の Jacobian を算出する [49].

例えば，$\sin(\cos(x))$ を計算する場合，次のようにする．

1. 変数値 x と微分係数 1 から計算を始める．
2. 中間的な関数値 $y = \cos x$ を算出するのに併せて微分係数 $y' = -\sin x$ を算出する．
3. 関数値 $z = \sin y$ を算出するのに併せて微分係数 $z' = y' \cos y$ を算出する．

この仕組みでは，各変数に対して微分係数用の領域を用意して計算を進めていく．

26.2 後退型自動微分

後退型自動微分では，式 (26.1) における $\boldsymbol{f}(\boldsymbol{y})$ の Jacobian が分かっている状態から $\boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))$ の Jacobian を算出する [49].

例えば，$\sin(\cos(x))$ を計算する場合，次のようにする．

1. 中間的な関数値 $y = \cos x$ を算出し，cos の計算をしたことを記録する．
2. 関数値 $z = \sin y$ を算出し，sin の計算をしたことを記録する．
3. 微分係数 1 から計算を始め，$1 \cdot dz/dy = \cos y$ を計算する．
4. $1 \cdot dz/dy \cdot dy/dx = \cos y \cdot (-\sin x)$ を計算する．

後退型自動微分においては，計算時とは逆順に微分係数の計算を行う必要がある．そこで，図 26.1 のような計算グラフと呼ばれるものを考える [49]．図 26.1 の中で上の方にあるノードから順に微分係数の計算を進めていく方法とし

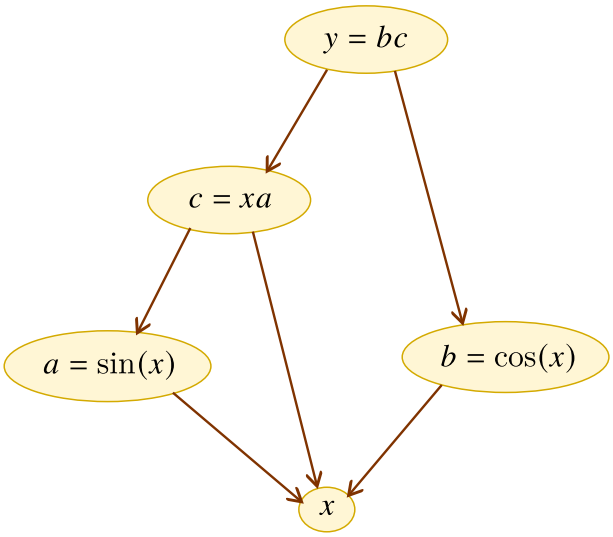


図 26.1: 後退型自動微分における $y = x \sin x \cos x$ の計算グラフの例

Algorithm 26.1 後退型自動微分における微分係数の計算

```
1:  procedure BackwardModeAutomaticDifferentiation(グラフ, 微分の対象となるノード)
2:      V, r = ListRequiredNodes(グラフ, 微分の対象となるノード)        ▷ Algorithm 26.2
3:      for all V の要素 v do
4:          d(v) ← 0                                                  ▷ 微分係数
5:          a(v) ← 0                                                  ▷ 微分係数の加算を行った回数
6:      end for
7:      d(微分の対象となるノード) ← 1
8:      空の処理待ちのキューを用意する.
9:      処理待ちのキューに微分の対象となるノードを追加する.
10:     while 処理待ちのキューに要素がある do
11:         処理待ちのキューから要素を取り出し，v とする.
12:         for all v が参照しているノード w do
13:             v を w で偏微分した係数 f を計算する.
14:             d(w) ← d(w) + f d(v)
15:             a(w) ← a(w) + 1
16:             if a(w) = r(w) then
17:                 w を処理待ちのキューに追加する.
18:             end if
19:         end for
20:     end while
21: end procedure
```

Algorithm 26.2 後退型自動微分における微分係数の計算におけるサブルーチン ListRequiredNodes

```
1: procedure ListRequiredNodes(グラフ, 微分の対象となるノード)
2:     V ← {微分の対象となるノード}                    ▷ 必要なノードの一覧
3:     r(微分の対象となるノード) ← 0                    ▷ ノードが参照されている回数
4:     空の処理待ちのキューを用意する.
5:     処理待ちのキューに微分の対象となるノードを追加する.
6:     while 処理待ちのキューに要素がある do
7:         処理待ちのキューから要素を取り出し，v とする.
8:         for all v が参照しているノード w do
9:             if r(w) が定義されている then
10:                r(w) ← r(w) + 1
11:            else
12:                r(w) ← 1
13:            end if
14:            if w が V にない then
15:                V ← V + {w}
16:                w を処理待ちのキューに追加する.
17:            end if
18:        end for
19:    end while
20:    return V, r
21: end procedure
```

ては，例えば Algorithm 26.1 が考えられる[*1].

[*1] 文献 [49] には実装に使用できるレベルのアルゴリズムが見当たらなかったため，トポロジカルソートをベースに考案した.

第 VII 部

正則化

第 27 章

導入

この章では，正則化のアルゴリズムをまとめる．

未知のパラメータ $\boldsymbol{x}$ に合わせて変動するデータ $\boldsymbol{y} = A\boldsymbol{x}$ をもとにパラメータ $\boldsymbol{x}$ を推定する問題は一般に 逆問題と呼ばれる．ここで，$\boldsymbol{x}, \boldsymbol{y}$ はそれぞれ何らかのノルム空間 X, Y に存在するベクトルで，A は何らかの作用素とする．例えば，録音した音源をもとに音が鳴っている場所を調べたい場合，音が鳴っている場所が $\boldsymbol{x}$ であり，録音した音源は $\boldsymbol{y}$ になる．

逆問題を解く単純な手法として，ノルム $\|A\boldsymbol{x} - \boldsymbol{y}\|$ の最小化（最小二乗法）が挙げられるが，作用素 A の性質によっては，$\boldsymbol{y}$ が少し変動するだけで最小二乗法による解 $\boldsymbol{x}$ が大きく変動してしまう場合があったり，解が複数存在してしまう場合があったりする（ ill-posed problem と呼ばれる）．そのような場合に，追加の情報をもとに唯一の安定な解を求められるようにする手法を正則化と呼ぶ．ここでは，特に次のような式を最小化する正則化手法における数値解法をまとめる．

$$\|A\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda R(\boldsymbol{x}) \tag{27.1}$$

ここで，$\lambda \in [0, \infty)$ は正則化パラメータと呼ばれるパラメータで，R は $R : X \to [0, \infty)$ のような関数である．この式を最小化する解 $\boldsymbol{x}_\lambda$ は正則化パラメータ λ により変化する．$\lambda = 0$ では正則化の効果がなくなり，λ を大きくすると正則化の効果が強くなる．λ を大きくしすぎると残差 $\|A\boldsymbol{x} - \boldsymbol{y}\|^2$ が大きくなり，データ $\boldsymbol{y}$ から離れていってしまうため，正則化パラメータ λ を適切に調整することが必要となる．

第 28 章

Tikhonov 正則化

ここでは基本的な Tikhonov 正則化についてまとめる．Tikhonov 正則化では，評価関数

$$E_\lambda(\boldsymbol{x}) \equiv \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2 \tag{28.1}$$

を最小化する．ここで，$\boldsymbol{x} \in \mathbb{C}^n$, $\boldsymbol{y} \in \mathbb{C}^m$, $A \in \mathbb{C}^{m\times n}$ である．

28.1　係数行列による解

評価関数 (28.1) を最小化する $\boldsymbol{x}$ は係数行列 A により次のように表すことができる．

定理 28.1. $\lambda > 0$ の場合，評価関数 (28.1) が最小となるのは，$\boldsymbol{x} = (A^*A + \lambda I)^{-1}A^*\boldsymbol{y}$ の場合である．

証明. この場合，ノルムを展開することで，次のようになる．

$$\begin{aligned} E_\lambda(\boldsymbol{x}) &= \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2 \\ &= (A\boldsymbol{x} - \boldsymbol{y})^*(A\boldsymbol{x} - \boldsymbol{y}) + \lambda\boldsymbol{x}^*\boldsymbol{x} \\ &= \boldsymbol{x}^*(A^*A + \lambda I)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \|\boldsymbol{y}\|_2^2 \end{aligned} \tag{28.2}$$

ここで，$\lambda > 0$ の場合は $\boldsymbol{x} \neq \boldsymbol{0}$ において $\boldsymbol{x}^*(A^*A + \lambda I)\boldsymbol{x} = \|A\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2 > 0$ となることから，エルミート行列 $(A^*A + \lambda I)$ は正定値であり，正則となる．このことを用いると，さらに次のように展開できる．

$$\begin{aligned} E_\lambda(\boldsymbol{x}) &= \boldsymbol{x}^*(A^*A + \lambda I)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \|\boldsymbol{y}\|_2^2 \\ &= \boldsymbol{x}^*(A^*A + \lambda I)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \boldsymbol{y}^*A(A^*A + \lambda I)^{-1}A^*\boldsymbol{y} - \boldsymbol{y}^*A(A^*A + \lambda I)^{-1}A^*\boldsymbol{y} + \|\boldsymbol{y}\|_2^2 \\ &= (\boldsymbol{x} - (A^*A + \lambda I)^{-1}A^*\boldsymbol{y})^*(A^*A + \lambda I)(\boldsymbol{x} - (A^*A + \lambda I)^{-1}A^*\boldsymbol{y}) - \boldsymbol{y}^*A(A^*A + \lambda I)^{-1}A^*\boldsymbol{y} + \|\boldsymbol{y}\|_2^2 \end{aligned} \tag{28.3}$$

エルミート行列 $(A^*A + \lambda I)$ が正定値であることから，この式が最小となるのは $(\boldsymbol{x} - (A^*A + \lambda I)^{-1}A^*\boldsymbol{y})$ が零ベクトルとなる場合，つまり，$\boldsymbol{x} = (A^*A + \lambda I)^{-1}A^*\boldsymbol{y}$ の場合である． □

ここで，$\lambda = 0$ でも，行列 A のランクが n の場合は A^*A が正定値になるため同様に解が求まる．

28.2　特異値分解による解法

行列 A を次のように特異値分解する．

$$A = U\begin{pmatrix} \Sigma & O \\ O & O \end{pmatrix}V^* \tag{28.4}$$

ここで，$U \in \mathbb{C}^{m\times m}$ と $V \in \mathbb{C}^{n\times n}$ はユニタリ行列で，$\Sigma \in \mathbb{R}^{r\times r}$ は正の実数による対角行列である．ただし，ランク r は m または n に等しくても良い．

この分解を用いると，定理 28.1 の解は次のように変形できる．

$$\begin{aligned}
x_\lambda &\equiv (A^*A + \lambda I)^{-1}A^*\boldsymbol{y} \\
&= \left(V\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}U^*U\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}V^* + \lambda I\right)^{-1} V\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y} \\
&= \left(V\begin{pmatrix}\Sigma^2 & O\\ O & O\end{pmatrix}V^* + \lambda VV^*\right)^{-1} V\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y} \\
&= \left(V\begin{pmatrix}\Sigma^2+\lambda I & O\\ O & \lambda I\end{pmatrix}V^*\right)^{-1} V\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y} \\
&= V\begin{pmatrix}(\Sigma^2+\lambda I)^{-1} & O\\ O & \lambda^{-1}I\end{pmatrix}V^*V\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y} \\
&= V\begin{pmatrix}(\Sigma^2+\lambda I)^{-1}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y}
\end{aligned} \tag{28.5}$$

さらに，$U = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)$, $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ とすると，次のように表記できる．

$$x_\lambda = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda}(\boldsymbol{u}_i^*\boldsymbol{y})\boldsymbol{v}_i \tag{28.6}$$

特異値分解を一回行い $\boldsymbol{u}_i^*\boldsymbol{y}$ を計算しておけば，ある正則化パラメータ λ に対する解 $\boldsymbol{x}_\lambda$ は単純な線形和で求めることができる．

また，$U_1 \equiv (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r)$, $U_2 \equiv (\boldsymbol{u}_{r+1}, \boldsymbol{u}_{r+2}, \ldots, \boldsymbol{u}_m)$ と定義した場合，U がユニタリ行列であることから $U_1^*U_1 = I$, $U_1^*U_2 = O$, $U_2^*U_1 = O$, $U_2^*U_2 = I$, $U_1U_1^* + U_2U_2^* = I$ が成り立つことを利用すると，正則化パラメータを評価する際にしばしば利用される評価関数内のノルムは次のように計算できる[*1]．

$$\begin{aligned}
\|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|_2^2 &= \left\|U\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}V^*V\begin{pmatrix}(\Sigma^2+\lambda I)^{-1}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y} - \boldsymbol{y}\right\|_2^2 \\
&= \left\|\begin{pmatrix}U_1 & U_2\end{pmatrix}\begin{pmatrix}\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma & O\\ O & O\end{pmatrix}\begin{pmatrix}U_1^*\\ U_2^*\end{pmatrix}\boldsymbol{y} - \boldsymbol{y}\right\|_2^2 \\
&= \left\|U_1\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma U_1^*\boldsymbol{y} - \boldsymbol{y}\right\|_2^2 \\
&= \left\|U_1\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma U_1^*\boldsymbol{y} - U_1U_1^*\boldsymbol{y} - U_2U_2^*\boldsymbol{y}\right\|_2^2 \\
&= \left\|U_1(\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma - I)U_1^*\boldsymbol{y} - U_2U_2^*\boldsymbol{y}\right\|_2^2 \\
&= \left\|U_1(\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma - I)U_1^*\boldsymbol{y}\right\|_2^2 + \left\|U_2U_2^*\boldsymbol{y}\right\|_2^2 \\
&= \left\|(\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma - I)U_1^*\boldsymbol{y}\right\|_2^2 + \left\|U_2U_2^*\boldsymbol{y}\right\|_2^2 \\
&= \sum_{i=1}^{r}\left(\frac{\lambda}{\sigma_i^2+\lambda}\right)^2(\boldsymbol{u}_i^*\boldsymbol{y})^2 + \left\|(I - U_1U_1^*)\boldsymbol{y}\right\|_2^2
\end{aligned} \tag{28.7}$$

$$\|\boldsymbol{x}_\lambda\|_2^2 = \left\|V\begin{pmatrix}(\Sigma^2+\lambda I)^{-1}\Sigma & O\\ O & O\end{pmatrix}U^*\boldsymbol{y}\right\|_2^2$$

[*1] 行列 U のうち U_1 の部分だけを求めた方が計算コストを抑えられるため，U_2 はなるべく使用しない計算式を求めている．

$$
\begin{aligned}
&= \left\| \begin{pmatrix} (\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \boldsymbol{y} \right\|_2^2 \\
&= \sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_i^2 + \lambda} \right)^2 (\boldsymbol{u}_i^* \boldsymbol{y})^2
\end{aligned} \tag{28.8}
$$

どちらも正則化パラメータごとに異なる部分は $O(r)$ オーダーで計算できる．

特異値分解による Tikhonov 正則化では，正則化パラメータを変更した際の再計算が容易なため，多数の正則化パラメータを試したい場合に便利である．

28.3　係数行列が横長の場合

行列 $A \in \mathbb{C}^{m \times n}$ が横長，つまり $m \ll n$ が成り立つ場合[*2]，定理 28.1 の解を

$$
\begin{aligned}
\boldsymbol{x} &= (A^*A + \lambda I)^{-1} A^* \boldsymbol{y} \\
&= A^* (AA^* + \lambda I)^{-1} \boldsymbol{y}
\end{aligned} \tag{28.9}
$$

のように変形することで，逆行列の計算の対象となる行列を $(A^*A + \lambda I) \in \mathbb{C}^{n \times n}$ から $(AA^* + \lambda I) \in \mathbb{C}^{m \times m}$ へ小さくできる．この変形は，$n \to \infty$ の場合を考えるのに便利である．

この変換を導出するには，次のような性質を用いる．

補題 28.1 ([50, p.18,19] より和訳)**.** A, R を正則行列とし，X, Y は $B = A + XRY$ が計算できるようなサイズの行列であるとする．B が正則であれば次の式が成り立つ．

$$
B^{-1} = A^{-1} - A^{-1}X(R^{-1} + YA^{-1}X)^{-1}YA^{-1} \tag{28.10}
$$

これを用いると，次のように式 (28.9) が示せる．

$$
\begin{aligned}
\boldsymbol{x} &= (A^*A + \lambda I)^{-1} A^* \boldsymbol{y} \\
&= \frac{1}{\lambda} \left(I + A^* \frac{1}{\lambda} IA \right)^{-1} A^* \boldsymbol{b} \\
&= \frac{1}{\lambda} \left(I - IA^* \left(\lambda I + AIA^* \right)^{-1} AI \right) A^* \boldsymbol{y} \\
&= \frac{1}{\lambda} \left(A^* - A^* \left(AA^* + \lambda I \right)^{-1} AA^* \right) \boldsymbol{y} \\
&= \frac{1}{\lambda} A^* \left(I - (AA^* + \lambda I)^{-1} AA^* \right) \boldsymbol{y} \\
&= \frac{1}{\lambda} A^* \left((AA^* + \lambda I)^{-1} (AA^* + \lambda I) - (AA^* + \lambda I)^{-1} AA^* \right) \boldsymbol{y} \\
&= \frac{1}{\lambda} A^* (AA^* + \lambda I)^{-1} (\lambda I) \boldsymbol{y} \\
&= A^* (AA^* + \lambda I)^{-1} \boldsymbol{y}
\end{aligned} \tag{28.11}
$$

なお，エルミート行列 AA^* は行列 A に依らず半正定値だから，正定値の λI を足した $(AA^* + \lambda I)$ は A に依らず正定値で逆行列を持つ．

[*2] 劣決定 (underdetermined) と呼ばれる．

第 29 章

一般化 Tikhonov 正則化

前章の Tikhonov 正則化では，正則化項が $\|\boldsymbol{x}\|_2^2$ となっていたが，正則化項として係数行列を追加した $\|L\boldsymbol{x}\|_2^2$ を用いることも考えられる．例えば，変数 $\boldsymbol{x}$ の隣り合う要素の差をとるように係数行列 L を決めることにより，解が滑らかになるような正則化を行うことができる．このような一般化した Tikhonov 正則化では，評価関数

$$E_\lambda(\boldsymbol{x}) \equiv \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|L\boldsymbol{x}\|_2^2 \tag{29.1}$$

を最小化する．ここで，$\boldsymbol{x} \in \mathbb{C}^n$, $\boldsymbol{y} \in \mathbb{C}^m$, $A \in \mathbb{C}^{m\times n}$, $L \in \mathbb{C}^{p\times n}$ である．

ここで，正則化項の変更により注意することがある．Tikhonov 正則化では，$\lambda > 0$ であれば解が唯一となっていた（定理 28.1）．しかし，一般化 Tikhonov 正則化においては，$\lambda > 0$ であるからといって解が唯一になるとは限らない．

定理 29.1. A と L の核空間の共通部分が $\mathbf{0}$ 以外の要素を持つ場合，$\lambda > 0$ であったとしても評価関数 (29.1) が最小となる $\boldsymbol{x}$ が唯一に定まらない．

証明. 評価関数 (29.1) が最小となる $\boldsymbol{x}$ の 1 つを $\bar{\boldsymbol{x}}$ とする．また，A と L の核空間の共通部分が $\mathbf{0}$ 以外に持つ要素の 1 つを $\boldsymbol{x}_0$ とする．このとき，

$$\begin{aligned} E_\lambda(\bar{\boldsymbol{x}} + \boldsymbol{x}_0) &= \|A(\bar{\boldsymbol{x}} + \boldsymbol{x}_0) - \boldsymbol{y}\|_2^2 + \lambda\|L(\bar{\boldsymbol{x}} + \boldsymbol{x}_0)\|_2^2 \\ &= \|A\bar{\boldsymbol{x}} - \boldsymbol{y}\|_2^2 + \lambda\|L\bar{\boldsymbol{x}}\|_2^2 \\ &= E(\bar{\boldsymbol{x}}) \end{aligned} \tag{29.2}$$

となる．よって，評価関数 $E(\boldsymbol{x})$ が最小となる $\boldsymbol{x}$ は唯一に定まらない． □

一方，A と L の核空間の共通部分が $\mathbf{0}$ のみであれば $\lambda > 0$ のときに解が唯一となる．

定理 29.2. A と L の核空間の共通部分が $\mathbf{0}$ のみでかつ，$\lambda > 0$ の場合，評価関数 (29.1) が最小となるのは，$\boldsymbol{x} = (A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y}$ の場合である．

証明. この場合，ノルムを展開することで，次のようになる．

$$\begin{aligned} E_\lambda(\boldsymbol{x}) &= \|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|L\boldsymbol{x}\|_2^2 \\ &= (A\boldsymbol{x} - \boldsymbol{y})^*(A\boldsymbol{x} - \boldsymbol{y}) + \lambda\boldsymbol{x}^*L^*L\boldsymbol{x} \\ &= \boldsymbol{x}^*(A^*A + \lambda L^*L)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \|\boldsymbol{y}\|_2^2 \end{aligned} \tag{29.3}$$

ここで，A と L の核空間の共通部分が $\mathbf{0}$ のみでかつ $\lambda > 0$ の場合は $\boldsymbol{x} \neq \mathbf{0}$ において $\boldsymbol{x}^*(A^*A + \lambda L^*L)\boldsymbol{x} = \|A\boldsymbol{x}\|_2^2 + \lambda\|L\boldsymbol{x}\|_2^2 > 0$ となることから，エルミート行列 $(A^*A + \lambda I)$ は正定値であり，正則となる．このことを用いると，さらに次のように展開できる．

$$E_\lambda(\boldsymbol{x}) = \boldsymbol{x}^*(A^*A + \lambda L^*L)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \|\boldsymbol{y}\|_2^2$$

$$
\begin{aligned}
&= \boldsymbol{x}^*(A^*A + \lambda L^*L)\boldsymbol{x} - \boldsymbol{x}^*A^*\boldsymbol{y} - \boldsymbol{y}^*A\boldsymbol{x} + \boldsymbol{y}^*A(A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y} - \boldsymbol{y}^*A(A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y} + \|\boldsymbol{y}\|_2^2 \\
&= (\boldsymbol{x} - (A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y})^*(A^*A + \lambda L^*L)(\boldsymbol{x} - (A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y}) - \boldsymbol{y}^*A(A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y} + \|\boldsymbol{y}\|_2^2
\end{aligned} \tag{29.4}
$$

エルミート行列 $(A^*A + \lambda L^*L)$ が正定値であることから，この式が最小となるのは $(\boldsymbol{x} - (A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y})$ が零ベクトルとなる場合，つまり，$\boldsymbol{x} = (A^*A + \lambda L^*L)^{-1}A^*\boldsymbol{y}$ の場合である．□

29.1　一般化特異値分解

$m \geq n \geq p$ の場合は次のように表される一般化特異値分解を行うことができる [51].

$$
A = U\begin{pmatrix} \Sigma & O \\ O & I \end{pmatrix}W^{-1}, \qquad L = V\begin{pmatrix} M & O \end{pmatrix}W^{-1} \tag{29.5}
$$

ここで，$U \in \mathbb{C}^{m\times n}$, $V \in \mathbb{C}^{p\times p}$ はユニタリ行列で，$W \in \mathbb{C}^{n\times n}$ は正則行列とし，$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$, $M = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p)$ は対角行列である．σ_i と μ_i については $0 \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_p \leq 1$, $1 \geq \mu_1 \geq \mu_2 \geq \ldots \geq \mu_p > 0$, $\sigma_i^2 + \mu_i^2 = 1$ を満たすものとし，$\gamma_i = \sigma_i/\mu_i$ を一般化特異値と呼ぶ．これを用いると，次のように評価関数 (29.1) を最小化する $\boldsymbol{x}$ を表せる [51].

$$
\boldsymbol{x}_\lambda = \sum_{i=1}^{p} \frac{\gamma_i/\mu_i}{\gamma_i^2 + \lambda}(\boldsymbol{u}_i^*\boldsymbol{y})\boldsymbol{w}_i + \sum_{i=p+1}^{n}(\boldsymbol{u}_i^*\boldsymbol{y})\boldsymbol{w}_i \tag{29.6}
$$

$L = I$ のときは，$p = n$, $\mu_i = 1$ となり，Tikhonov 正則化の場合の式 (28.6) に一致する．

29.2　単純な L2 ノルムによる Tikhonov 正則化への変換

一般化 Tikhonov 正則化を通常の Tikhonov 正則化へ変形することもできる [51]．その変形では，次のように変数を置き換える．

$$
\boldsymbol{x} = L_A^\dagger\bar{\boldsymbol{x}} + \boldsymbol{x}_0, \qquad L_A^\dagger = \left(I - (AP_L)^\dagger A\right)L^\dagger, \qquad \boldsymbol{x}_0 = (AP_L)^\dagger\boldsymbol{y} \tag{29.7}
$$

ただし，$P_L = I - L^\dagger L$ とする．一般化 Tikhonov 正則化の問題を数値的に解く際はこの変換で通常の Tikhonov 正則化に直して解くと安定かつ効率的に解が得られることが知られている [51]．そこで，この変換について以下に示す．

まず，式 (29.7) の変換は次のような意味を持っている．

補題 29.1 ([52, 3 節]). 最適化問題

$$
\begin{aligned}
\text{minimize} \quad & \|A\boldsymbol{x} - \boldsymbol{y}\|_2 \\
\text{s.t.} \quad & L\boldsymbol{x} = \bar{\boldsymbol{x}}
\end{aligned}
$$

において，行列 $A \in \mathbb{C}^{m\times n}$, $L \in \mathbb{C}^{p\times n}$（$m \geq n \geq p$）は核空間に $\boldsymbol{0}$ 以外の共通部分を持たないものとする．このとき，この問題の解は

$$
\boldsymbol{x} = L_A^\dagger\bar{\boldsymbol{x}} + (AP_L)^\dagger\boldsymbol{y} \tag{29.8}
$$

である．

証明. まず，$\boldsymbol{Lx} = \bar{\boldsymbol{x}}$ の一般解は次のように書ける（式 (4.1)）.

$$\boldsymbol{x} = L^{\dagger}\bar{\boldsymbol{x}} + P_L \boldsymbol{z} \tag{29.9}$$

これを目的関数に代入すると

$$\|A\boldsymbol{x} - \boldsymbol{y}\|_2 = \|AL^{\dagger}\bar{\boldsymbol{x}} + AP_L\boldsymbol{z} - \boldsymbol{y}\|_2 = \|(AP_L)\boldsymbol{z} - (\boldsymbol{y} - AL^{\dagger}\bar{\boldsymbol{x}})\|_2 \tag{29.10}$$

のように変形でき，このノルムを最小化する $\boldsymbol{z}$ は次のように書ける.

$$\boldsymbol{z} = (AP_L)^{\dagger}(\boldsymbol{y} - AL^{\dagger}\bar{\boldsymbol{x}}) + (I - (AP_L)^{\dagger}(AP_L))\boldsymbol{s} \tag{29.11}$$

ここで，$\boldsymbol{s}$ は任意のベクトルである．ここで，P_L は L の核空間への射影演算子であり，A と L が核空間に $\boldsymbol{0}$ 以外の共通部分を持たないことから，AP_L の核空間は L の核空間の直交補空間である．任意のベクトル $\boldsymbol{a}$ について $(AP_L)^{\dagger}\boldsymbol{a}$ は AP_L の核空間の成分を持たないため，L の核空間に属していることが分かる．一方，$(I - (AP_L)^{\dagger}(AP_L))\boldsymbol{s}$ は AP_L の核空間に属しており，L の核空間に直交する．よって，

$$P_L\boldsymbol{z} = (AP_L)^{\dagger}(\boldsymbol{y} - AL^{\dagger}\bar{\boldsymbol{x}}) \tag{29.12}$$

となる.

このときの $\boldsymbol{x}$ は

$$\begin{aligned}
\boldsymbol{x} &= L^{\dagger}\bar{\boldsymbol{x}} + P_L\boldsymbol{z} \\
&= L^{\dagger}\bar{\boldsymbol{x}} + (AP_L)^{\dagger}(\boldsymbol{y} - AL^{\dagger}\bar{\boldsymbol{x}}) \\
&= \left(I - (AP_L)^{\dagger}A\right)L^{\dagger}\bar{\boldsymbol{x}} + (AP_L)^{\dagger}\boldsymbol{y} \\
&= L_A^{\dagger}\bar{\boldsymbol{x}} + (AP_L)^{\dagger}\boldsymbol{y}
\end{aligned} \tag{29.13}$$

となり，解が得られる. □

この補題を用いることで，次のように評価関数の変換を示すことができる.

定理 29.3 ([51]). A と L の核空間の共通部分が $\boldsymbol{0}$ のみである場合，変数変換 (29.7) により評価関数 (29.1) を次のように置き換えることができる.

$$E'(\bar{\boldsymbol{x}}) = \left\|\bar{A}\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}\right\|_2^2 + \lambda\|\bar{\boldsymbol{x}}\|_2^2 \tag{29.14}$$

ただし，

$$\bar{A} = AL_A^{\dagger} \tag{29.15}$$

$$\bar{\boldsymbol{y}} = \boldsymbol{y} - A\boldsymbol{x}_0 \tag{29.16}$$

$$\boldsymbol{x}_0 = (AP_L)^{\dagger}\boldsymbol{y} \tag{29.17}$$

である.

証明. 補題 29.1 より，次のように変形できる.

$$\begin{aligned}
\min_{\boldsymbol{x}\in\mathbb{C}^n} E(\boldsymbol{x}) &= \min_{\boldsymbol{x}\in\mathbb{C}^n}\left(\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|L\boldsymbol{x}\|_2^2\right) \\
&= \min_{\bar{\boldsymbol{x}}\in\mathbb{C}^p}\min_{\boldsymbol{x}\in\{\boldsymbol{x}\in\mathbb{C}^n | L\boldsymbol{x}=\bar{\boldsymbol{x}}\}}\left(\|A\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|L\boldsymbol{x}\|_2^2\right) \\
&= \min_{\bar{\boldsymbol{x}}\in\mathbb{C}^p}\left(\left\|A\left(L_A^{\dagger}\bar{\boldsymbol{x}} + \boldsymbol{x}_0\right) - \boldsymbol{y}\right\|_2^2 + \lambda\,\|\bar{\boldsymbol{x}}\|_2^2\right)
\end{aligned}$$

$$
\begin{aligned}
&= \min_{\bar{\boldsymbol{x}} \in \mathbb{C}^p} \left(\left\| \bar{A}\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}} \right\|_2^2 + \lambda \left\| \bar{\boldsymbol{x}} \right\|_2^2 \right) \\
&= \min_{\bar{\boldsymbol{x}} \in \mathbb{C}^p} E'(\bar{\boldsymbol{x}})
\end{aligned} \tag{29.18}
$$

以上より，$E(\boldsymbol{x})$ の最小化は $E'(\bar{\boldsymbol{x}})$ の最小化へ置き換えることができる． □

最後にこの変換をコンピュータ上で行う手法について文献 [52, 53] の結果をまとめる．

29.2.1　行列 L が行と同じ数のランクを持つ場合の計算方法

ここでは，$L \in \mathbb{C}^{p \times n}$ が $p < n$ でランク p である場合を扱う．

まず，L^* を QR 分解する．

$$
L^* = \begin{pmatrix} V_1 & V_2 \end{pmatrix} \begin{pmatrix} R \\ O \end{pmatrix} \tag{29.19}
$$

ここで，$V_1 \in \mathbb{C}^{n \times p}$, $V_2 \in \mathbb{C}^{n \times (n-p)}$ は $V = (V_1, V_2)$ がユニタリ行列になるような行列とし，$R \in \mathbb{C}^{p \times p}$ は正則な上三角行列である．このとき，$L^\dagger = V_1 R^{-*}$, $LV_2 = O$ となることと，A と L の核空間の共通部分が $\boldsymbol{0}$ のみであることから，AV_2 は列と同じ数のランクを持つ．よって，次のように QR 分解できる．

$$
AV_2 = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} U \\ O \end{pmatrix} \tag{29.20}
$$

これも同様に $Q_1 \in \mathbb{C}^{m \times (n-p)}$, $Q_2 \in \mathbb{C}^{m \times (m-n+p)}$ は $Q = (Q_1, Q_2)$ がユニタリ行列になるような行列とし，$U \in \mathbb{C}^{(n-p) \times (n-p)}$ は正則な上三角行列である．このとき，

$$
\begin{aligned}
\boldsymbol{x}_0 &= \left(A(I - L^\dagger L) \right)^\dagger \boldsymbol{y} \\
&= \left(AV_2 V_2^* \right)^\dagger \boldsymbol{y}
\end{aligned} \tag{29.21}
$$

となる．

ここで，$(AV_2V_2^*)^\dagger$ について定義に沿って考える．最適化問題

$$
\begin{aligned}
\text{minimize} \quad & \|\boldsymbol{x}\|_2 \\
\text{s.t.} \quad & \|AV_2V_2^*\boldsymbol{x} - \boldsymbol{a}\|_2 = \min_{\boldsymbol{x}} \|AV_2V_2^*\boldsymbol{x} - \boldsymbol{a}\|_2
\end{aligned}
$$

において，$\boldsymbol{x} = V_1\boldsymbol{s}_1 + V_2\boldsymbol{s}_2$ とおくと，$\|AV_2V_2^*\boldsymbol{x} - \boldsymbol{a}\|_2 = \|AV_2\boldsymbol{s}_2 - \boldsymbol{a}\|_2$ となる．これを最小化すると $\boldsymbol{s}_2 = (AV_2)^\dagger \boldsymbol{a}$ となる．また，$\|\boldsymbol{x}\|_2^2 = \|\boldsymbol{s}_1\|_2^2 + \|\boldsymbol{s}_2\|_2^2$ となるから，$\boldsymbol{s}_2 = (AV_2)^\dagger \boldsymbol{a}$ は固定された状態で $\|\boldsymbol{x}\|_2$ を最小化すると $\boldsymbol{s}_1 = \boldsymbol{0}$ となる．よって，この最適化問題の解は $\boldsymbol{x} = V_2(AV_2)^\dagger \boldsymbol{a}$ となり，$(AV_2V_2^*)^\dagger = V_2(AV_2)^\dagger$ とわかる．

よって，

$$
\begin{aligned}
\boldsymbol{x}_0 &= \left(A(I - L^\dagger L) \right)^\dagger \boldsymbol{y} \\
&= V_2 \left(AV_2 \right)^\dagger \boldsymbol{y} \\
&= V_2 U^{-1} Q_1^* \boldsymbol{y}
\end{aligned} \tag{29.22}
$$

となる．また，

$$
\begin{aligned}
\bar{\boldsymbol{y}} &= \boldsymbol{y} - A\boldsymbol{x}_0 \\
&= \boldsymbol{y} - AV_2U^{-1}Q_1^*\boldsymbol{y} \\
&= (I - Q_1Q_1^*)\boldsymbol{y} \\
&= Q_2Q_2^*\boldsymbol{y}
\end{aligned} \tag{29.23}
$$

と書ける．さらに，

$$\begin{aligned} L_A^\dagger &= \left(I - \left(A(I - L^\dagger L)\right)^\dagger A\right) L^\dagger \\ &= (I - V_2 U^{-1} Q_1^* A) V_1 R^{-*} \end{aligned} \tag{29.24}$$

となるから，

$$\begin{aligned} \bar{A} &= A L_A^\dagger \\ &= A(I - V_2 U^{-1} Q_1^* A) V_1 R^{-*} \\ &= (A - A V_2 U^{-1} Q_1^* A) V_1 R^{-*} \\ &= (I - A V_2 U^{-1} Q_1^*) A V_1 R^{-*} \\ &= (I - Q_1 U U^{-1} Q_1^*) A V_1 R^{-*} \\ &= (I - Q_1 Q_1^*) A V_1 R^{-*} \\ &= Q_2 Q_2^* A V_1 R^{-*} \end{aligned} \tag{29.25}$$

と変形できる．そして，$\bar{\boldsymbol{x}}$ から $\boldsymbol{x}$ への変換は

$$\begin{aligned} \boldsymbol{x} &= L_A^\dagger \bar{\boldsymbol{x}} + \boldsymbol{x}_0 \\ &= (I - V_2 U^{-1} Q_1^* A) V_1 R^{-*} \bar{\boldsymbol{x}} + V_2 U^{-1} Q_1^* \boldsymbol{y} \\ &= (I - V_2 U^{-1} Q_1^* A) L^\dagger \bar{\boldsymbol{x}} + V_2 U^{-1} Q_1^* \boldsymbol{y} \\ &= L^\dagger \bar{\boldsymbol{x}} + V_2 U^{-1} Q_1^* (\boldsymbol{y} - A L^\dagger \bar{\boldsymbol{x}}) \end{aligned} \tag{29.26}$$

でできる．さらに，$Q_2^* Q_2 = I$ を用いれば，

$$\left\| \bar{A}\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}} \right\|_2 = \left\| \tilde{A}\bar{\boldsymbol{x}} - \tilde{\boldsymbol{y}} \right\|_2, \qquad \tilde{A} = Q_2^* A V_1 R^{-*}, \qquad \tilde{\boldsymbol{y}} = Q_2^* \boldsymbol{y} \tag{29.27}$$

のように書き換えることもできる．

29.2.2　行列 L が正則な正方行列の場合の計算方法

行列 L が正則な場合，$L^\dagger = L^{-1}$ となるため，式がかなり単純になる．
まず，P_L は

$$P_L = I - L^\dagger L = I - L^{-1} L = O \tag{29.28}$$

となる．これを用いると，$L_A^\dagger$ は

$$L_A^\dagger = \left(I - (A P_L)^\dagger A\right) L^\dagger = \left(I - O^\dagger A\right) L^{-1} = (I - OA) L^{-1} = L^{-1} \tag{29.29}$$

となる．これらにより，変換の公式は以下のようになる．

$$\boldsymbol{x} = L^{-1} \bar{\boldsymbol{x}} \tag{29.30}$$

$$\bar{A} = A L^{-1} \tag{29.31}$$

$$\bar{\boldsymbol{y}} = \boldsymbol{y} \tag{29.32}$$

29.2.3　行列 L が一般の行列の場合の計算方法

まず，L を特異値分解する．

$$L = W \begin{pmatrix} \Omega & O \\ O & O \end{pmatrix} V^* \tag{29.33}$$

ここで，L はランク r で $W \in \mathbb{C}^{p\times p}$, $V \in \mathbb{C}^{n\times n}$ はユニタリ行列，$\Omega \in \mathbb{R}^{r\times r}$ は正数の対角成分による対角行列とする．このとき，$W = (W_1, W_2)$ なる行列 $W_1 \in \mathbb{C}^{p\times r}$ と $W_2 \in \mathbb{C}^{p\times(p-r)}$，$V = (V_1, V_2)$ なる行列 $V_1 \in \mathbb{C}^{n\times r}$ と $V_2 \in \mathbb{C}^{n\times(n-r)}$ を定義する．

このとき，L が列と同じ数のランクを持つ場合と同様にして

$$\boldsymbol{x}_0 = V_2 U^{-1} Q_1^* \boldsymbol{y} \tag{29.34}$$
$$\bar{\boldsymbol{y}} = Q_2 Q_2^* \boldsymbol{y} \tag{29.35}$$
$$L_A^\dagger = (I - V_2 U^{-1} Q_1^* A) V_1 \Omega^{-1} W_1^* \tag{29.36}$$
$$\bar{A} = Q_2 Q_2^* A V_1 \Omega^{-1} W_1^* \tag{29.37}$$
$$\tilde{A} = Q_2^* A V_1 \Omega^{-1} W_1^* \tag{29.38}$$
$$\tilde{\boldsymbol{y}} = Q_2^* \boldsymbol{y} \tag{29.39}$$

が得られる．

第 30 章

L-curve

ここでは正則化パラメータを決める手法の 1 つである L-curve 法について説明する．

正則化の式 (27.1) を最小化する $\boldsymbol{x}$ を $\boldsymbol{x}_\lambda$ として，横軸を残差 $\|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|$，縦軸を正則化項 $R(\boldsymbol{x}_\lambda)$ とし，正則化パラメータ λ を変えたときのプロットを L-curve と呼ぶ．L-curve は一般に図 30.1 のような L の字を描いている場合が多い．単純にノルムをプロットするよりも両対数グラフにプロットする方が後述する曲線の特徴をはっきりさせられる他，スケールの違いによる影響を抑えられるなどのメリットもあるという [51].

図 30.1 のように L-curve が得られた場合，L の字の曲がり角にあたる赤い点の部分から左上の方では残差がほとんど減らずに正則化項が増加しており，右下の方では正則化項がほとんど減らずに残差が増えているため，両方のバランスが取れている L の字の曲がり角を取るのが良いと考えられる．そこで，L-curve の曲がり角を数値的に求める手法について考える．

ここでは，文献 [51] に従い次の関数の組を考える．

$$(\xi(\lambda), \eta(\lambda)) = (\log \|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|, \log R(\boldsymbol{x})) \tag{30.1}$$

L-curve の曲がり角は曲線 $(\xi(\lambda), \eta(\lambda))$ の曲率が最も大きい部分と考えられるため，曲率

$$\kappa(\lambda) = \frac{\xi'\eta'' - \xi''\eta'}{\left((\xi')^2 + (\eta')^2\right)^{3/2}} \tag{30.2}$$

を求め，それを最大化する．曲率が何らかの手法で求まれば，1 変数の最適化については 11 章で説明しているため，ここでは曲率の計算法について考える．

30.1 Tikhonov 正則化の場合

Tikhonov 正則化（28 章）の場合，式 (30.2) にある各種微分を陽的に表すことができる [54, 55].

$$\frac{d}{d\lambda}\|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|_2^2 = \sum_{i=1}^{r} \frac{2\lambda\sigma_i^2}{(\sigma_i^2 + \lambda)^3} \left(\boldsymbol{u}_i^* \boldsymbol{y}\right)^2 \tag{30.3}$$

$$\frac{d}{d\lambda}\|\boldsymbol{x}_\lambda\|_2^2 = \sum_{i=1}^{r} \frac{-2\sigma_i^2}{(\sigma_i^2 + \lambda)^3} \left(\boldsymbol{u}_i^* \boldsymbol{y}\right)^2 \tag{30.4}$$

$$\frac{d^2}{d\lambda^2}\|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|_2^2 = \sum_{i=1}^{r} \frac{2\sigma_i^4 - 4\lambda\sigma_i^2}{(\sigma_i^2 + \lambda)^4} \left(\boldsymbol{u}_i^* \boldsymbol{y}\right)^2 \tag{30.5}$$

$$\frac{d^2}{d\lambda^2}\|\boldsymbol{x}_\lambda\|_2^2 = \sum_{i=1}^{r} \frac{6\sigma_i^2}{(\sigma_i^2 + \lambda)^4} \left(\boldsymbol{u}_i^* \boldsymbol{y}\right)^2 \tag{30.6}$$

これらは式 (28.7), (28.8) から求めることができる．

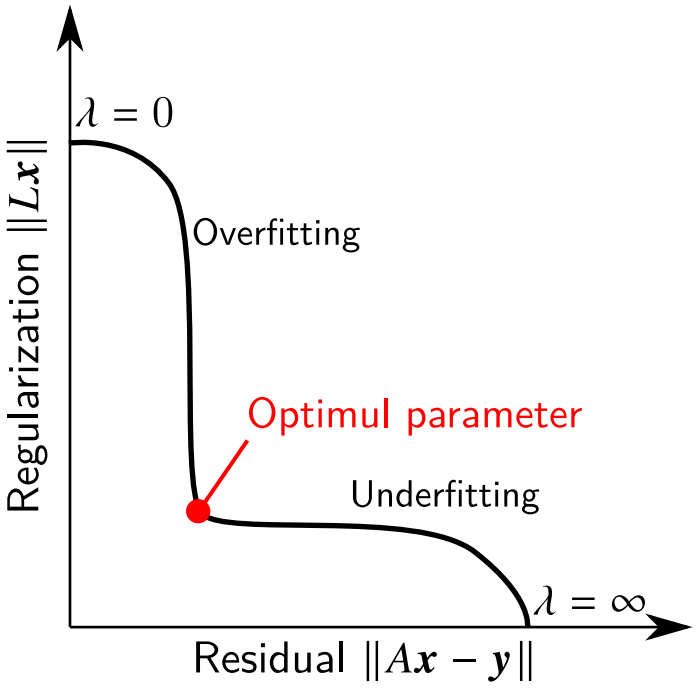


図 30.1: L-curve の概形

30.2 一般の場合

前節のように残差項と正則化項の微分を解析的に計算できれば簡単に曲率の計算ができるが，必ずしも正則化項を簡単に計算できるとは限らない．$\xi(\lambda)$ と $\eta(\lambda)$ の微分を解析的に計算できない場合は，サンプル点に対する 3 次の自然スプラインによる補間を用いて L-curve の曲率を計算できる．ただし，このときは $(\xi(\lambda), \eta(\lambda))$ を λ の関数として補間するのではなく，$(\lambda(t), \xi(\lambda(t)), \eta(\lambda(t)))$ のような媒介変数 t を用いた 3 次元データを補間しなければ上手くいかない [51, 56] [*1]．また，スプラインの計算において $(\xi(\lambda(t)), \eta(\lambda(t)))$ の近いサンプル点があると最終的な曲率に影響が出る場合があるため，そのような点を省いて計算を行う必要がある．

[*1] 例えば，サンプル点の距離を用いて長さのパラメータを作り，それを t にすることが考えられる．

第 31 章

Generalized Cross Validation

正則化パラメータを求めるための手法の 1 つに Generalized Cross Validation (GCV) がある．GCV では，次の式の最小化を行う [57].

$$V(\lambda) = \frac{\frac{1}{m}\|A\boldsymbol{x}_\lambda - \boldsymbol{y}\|^2}{\left|\frac{1}{m}\,\mathrm{tr}\{I - P_A(\lambda)\}\right|^2} \tag{31.1}$$

ここで，$\boldsymbol{x}_\lambda$ は正則化の式 (27.1) を最小化する $\boldsymbol{x}$ であり，$P_A(\lambda)$ は influence matrix と呼ばれるもので，データ $\boldsymbol{y}$ から解 $\boldsymbol{x}_\lambda$ を求める作用素を $A_\lambda^\#$ としたときに，

$$P_A(\lambda) \equiv \frac{\partial A A_\lambda^\# \boldsymbol{y}}{\partial \boldsymbol{y}} \tag{31.2}$$

のように定義される．

31.1 導出

まず，GCV の基になっている Ordinary Cross Validation (OCV) を文献 [57] に沿って示す．

OCV では次の関数を考える．

$$V_0(\lambda) \equiv \frac{1}{m}\sum_{i=1}^{m}\left|y_i - A_i\boldsymbol{x}_\lambda^{(i)}\right|^2 \tag{31.3}$$

ただし，$\boldsymbol{x}_\lambda^{(i)}$ は y_i 以外のデータから求めた解で，A_i は解からデータ y_i を推定する作用素である．

ここで次の補題が成り立つ．

補題 31.1 (文献 [57] の補題 4.2.1 より, leaving-out-one lemma)**.** 評価関数

$$\frac{1}{m}\left(|z - A_k\boldsymbol{x}|^2 + \sum_{i\neq k}|y_i - A_i\boldsymbol{x}|^2\right) + \lambda R(\boldsymbol{x}) \tag{31.4}$$

を最小化するような $\boldsymbol{x}$ を $\boldsymbol{h}_\lambda[k, z]$ とし，評価関数

$$\frac{1}{m}\sum_{i\neq k}|y_i - A_i\boldsymbol{x}|^2 + \lambda R(\boldsymbol{x}) \tag{31.5}$$

を最小化するような $\boldsymbol{x}$ を $\boldsymbol{x}_\lambda^{(k)}$ としたとき，$h_\lambda[k, A_k\boldsymbol{x}_\lambda^{(k)}] = \boldsymbol{x}_\lambda^{(k)}$ が成り立つ．

証明. $\tilde{y}_k = A_k \boldsymbol{x}_\lambda^{(k)}$ とし，$\boldsymbol{x} \neq \boldsymbol{x}_\lambda^{(k)}$ とする．このとき，

$$\begin{aligned}
&\frac{1}{m}\left(\left|\tilde{y}_k - A_k \boldsymbol{x}_\lambda^{(k)}\right|^2 + \sum_{i\neq k}\left|y_i - A_i \boldsymbol{x}_\lambda^{(k)}\right|^2\right) + \lambda R(\boldsymbol{x}_\lambda^{(k)}) \\
&= \frac{1}{m}\sum_{i\neq k}\left|y_i - A_i \boldsymbol{x}_\lambda^{(k)}\right|^2 + \lambda R(\boldsymbol{x}_\lambda^{(k)}) \\
&< \frac{1}{m}\sum_{i\neq k}|y_i - A_i \boldsymbol{x}|^2 + \lambda R(\boldsymbol{x}) \\
&\leq \frac{1}{m}\left(|\tilde{y}_k - A_k \boldsymbol{x}|^2 + \sum_{i\neq k}|y_i - A_i \boldsymbol{x}|^2\right) + \lambda R(\boldsymbol{x})
\end{aligned} \tag{31.6}$$

のように変形できる．よって，$h_\lambda[k, A_k \boldsymbol{x}_\lambda^{(k)}] = \boldsymbol{x}_\lambda^{(k)}$ である． □

これを用い，次の量を考える．

$$\tilde{p}_{kk}(\lambda) \equiv \frac{A_k \boldsymbol{x}_\lambda - A_k \boldsymbol{x}_\lambda^{(k)}}{y_k - A_k \boldsymbol{x}_\lambda^{(k)}} \tag{31.7}$$

まず，定義から，

$$y_k - A_k \boldsymbol{x}_\lambda^{(k)} = \frac{y_k - A_k \boldsymbol{x}_\lambda}{1 - \tilde{p}_{kk}(\lambda)} \tag{31.8}$$

が成り立つ．また，補題から

$$\tilde{p}_{kk}(\lambda) = \frac{A_k h[k, y_k] - A_k h[k, \tilde{y}_k]}{y_k - \tilde{y}_k} \tag{31.9}$$

が成り立つ．さらに，これは 1 次近似により

$$\tilde{p}_{kk}(\lambda) \approx \frac{\partial A_k h[k, y_k]}{\partial y_k} = \frac{\partial A_k \boldsymbol{x}_\lambda}{\partial y_k} = \frac{\partial A_k A_\lambda^{\#} \boldsymbol{y}}{\partial y_k} = p_{kk}(\lambda) \tag{31.10}$$

と書ける．ただし，$p_{kk}(\lambda)$ は influence matrix $P_A(\lambda)$ の (k,k) 成分である．

このことを利用して OCV の評価関数 (31.3) を変形すると，

$$V_0(\lambda) = \frac{1}{m}\sum_{i=1}^{m}\left|y_i - A_i \boldsymbol{x}_\lambda^{(i)}\right|^2 \approx \frac{1}{m}\sum_{i=1}^{m}\frac{|y_i - A_i \boldsymbol{x}_\lambda|^2}{|1 - p_{kk}(\lambda)|^2} \tag{31.11}$$

のように書ける．

$p_{kk}(\lambda)$ の代わりに，influence matrix の対角成分の平均値 $\operatorname{tr} P_A(\lambda)/n$ を用い，$\boldsymbol{y}$ の回転に影響されない評価関数を作ると，GCV の評価関数 (31.1) が得られる．

31.2　Tikhonov 正則化の場合における計算方法

Tikhonov 正則化（28 章）の場合，残差項は式 (28.7) で計算できる．そこで，influence matrix の計算について考える．

28 章と同様に行列 A の特異値分解

$$A = U\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}V^* \tag{31.12}$$

を行い，その場合の解（式 (28.5)）

$$x_\lambda = V \begin{pmatrix} (\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \boldsymbol{y} \tag{31.13}$$

を利用すると，

$$\begin{aligned} AA_\lambda^\# \boldsymbol{y} &= U \begin{pmatrix} \Sigma & O \\ O & O \end{pmatrix} V^* V \begin{pmatrix} (\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \boldsymbol{y} \\ &= U \begin{pmatrix} \Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \boldsymbol{y} \end{aligned} \tag{31.14}$$

となる．よって，influence matrix は

$$\begin{aligned} P_A(\lambda) &= \frac{\partial AA_\lambda^\# \boldsymbol{y}}{\partial \boldsymbol{y}} \\ &= U \begin{pmatrix} \Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \end{aligned} \tag{31.15}$$

であり，そのトレースは

$$\begin{aligned} \operatorname{tr} P_A(\lambda) &= \operatorname{tr}\left(U \begin{pmatrix} \Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* \right) \\ &= \operatorname{tr}\left(\begin{pmatrix} \Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} U^* U \right) \\ &= \operatorname{tr} \begin{pmatrix} \Sigma(\Sigma^2 + \lambda I)^{-1}\Sigma & O \\ O & O \end{pmatrix} \\ &= \sum_{i=1}^{r} \frac{\sigma_i^2}{\sigma_i^2 + \lambda} \end{aligned} \tag{31.16}$$

と計算できる [51] [*1]．

31.3　一般の場合における計算方法

解きたい方程式や正則化項が非線形な場合，反復的な最適化アルゴリズムで正則化した推定解を求めることになるが，そのような場合の GCV をどのように計算すれば良いのかについて，文献 [58] の結果を簡単に紹介する．なお，本節では，データ $\boldsymbol{y}$ について $\boldsymbol{y} \in \mathbb{R}^m$ を前提とする．

まず，非線形な問題においてどのような問題が起きるかをまとめる．式 (31.1) で示した GCV の評価関数において，分子は推定解 $\boldsymbol{x}_\lambda$ における残差であり，計算できる場合が多い．それに対して分母は推定解に A を作用した結果 $AA_\lambda^\# \boldsymbol{y}$ を $\boldsymbol{y}$ について偏微分したヤコビアン $P_A(\lambda)$ を含んでおり，非線形な問題においてはこれを解析的に計算することができない．数値的にこれを求めることも可能だが，その場合は全ての $i = 1, 2, \ldots, m$ について i 要素目だけ 1 の単位行列 $\boldsymbol{e}_i$（i の要素だけ 1 の単位ベクトル）の方向へデータをずらした $\boldsymbol{y} + \Delta y_i \boldsymbol{e}_i$ について推定解を得て A を作用させることにより，数値微分する必要がある．これは推定解を求めるのに時間のかかる最適化において現実的ではない．この問題を解決するために，文献 [58] では次のような 2 段階の近似を行っている．

まず，分母を次の式で近似する．

$$\frac{1}{m} \operatorname{tr}(I - P_A(\lambda)) \approx \frac{\boldsymbol{w}^T (I - P_A(\lambda)) \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \tag{31.17}$$

[*1] $\sigma_i^2/(\sigma_i^2 + \lambda)$ は filter factor と呼ばれる．

これは文献 [59] において提案された Monte-Carlo Cross-Validation と呼ばれる手法のもので，$\boldsymbol{w}$ は各要素を独立に平均 0，分散 1 の正規分布 $\mathcal{N}(0,1)$ からサンプリングしたベクトルである[*2]．この近似式については次のような定理が証明されている．

定理 31.1 (文献 [59] 定理 2.2)**.** 行列 $B \in \mathbb{R}^{m\times m}$ と各要素が独立に正規分布 $\mathcal{N}(0,1)$ に従うベクトル $\boldsymbol{w} \in \mathbb{R}^m$ を用いて

$$T_B^*(\boldsymbol{w}) \equiv \frac{\boldsymbol{w}^T B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \tag{31.18}$$

を定義する．このとき，T_B^* の平均と分散はそれぞれ

$$\mathrm{E}[T_B^*(\boldsymbol{w})] = \mu_1 \equiv \frac{1}{m}\operatorname{tr}(B) = \frac{1}{m}\sum_{i=1}^{m} B_{ii} \tag{31.19}$$

$$\mathrm{Var}[T_B^*(\boldsymbol{w})] = \frac{2}{m(m+2)}\sum_{i=1}^{m}(B_{ii}-\mu_1)^2 \tag{31.20}$$

となる．

ここで出てくる分散による誤差が気になる場合は，複数の $\boldsymbol{w}$ で式 (31.17) の近似を行った結果の平均で $\operatorname{tr}(I - P_A(\lambda))/n$ の推定を行うことで精度を上げることもできる．しかし，文献 [59] の数値実験では $m = 50$ でも真値と似たような挙動をとる推定値が得られており，$m = 500$ では真値との差をグラフから読み取るのが困難な程度に良い結果が得られている．

文献 [58] ではさらに

$$\begin{aligned} P_A(\lambda)\boldsymbol{w} &= \frac{\partial A A_\lambda^{\#}\boldsymbol{y}}{\partial \boldsymbol{y}}\boldsymbol{w} \\ &\approx \frac{A A_\lambda^{\#}(\boldsymbol{y}+h\boldsymbol{w}) - A A_\lambda^{\#}\boldsymbol{y}}{h} \end{aligned} \tag{31.21}$$

と 1 次近似することにより，次の式を得ている．

$$\begin{aligned} \frac{1}{m}\operatorname{tr}(I - P_A(\lambda)) &\approx \frac{\boldsymbol{w}^T(I - P_A(\lambda))\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^T}{\boldsymbol{w}^T\boldsymbol{w}}(\boldsymbol{w} - P_A(\lambda)\boldsymbol{w}) \\ &\approx \frac{\boldsymbol{w}^T}{\boldsymbol{w}^T\boldsymbol{w}}\left(\boldsymbol{w} - \frac{A A_\lambda^{\#}(\boldsymbol{y}+h\boldsymbol{w}) - A A_\lambda^{\#}\boldsymbol{y}}{h}\right) \end{aligned} \tag{31.22}$$

数値微分のずらす幅を決める h は $AA_\lambda^{\#}$ の演算の精度を見ながら調整することになるが，これにより推定解の計算を最低 2 回行えば GCV の評価関数を計算できる．

[*2] この手法は線形（つまり $P_A(\lambda) = AA_\lambda^{\#}$ と行列で書ける）だが大規模な問題を解くために考えられたもので，そのような問題では，たとえ $P_A(\lambda)$ が陽的に書けても，トレース $\operatorname{tr}(I - P_A(\lambda))$ を計算するのはかなりコストを要するという問題があり，このような手法が考案された．

第 VIII 部

補間

第 32 章

導入

この部では，補間の手法をまとめる．補間では，サンプル点 $(\boldsymbol{r}_i, y_i) \in X \times \mathbb{C}$ $(i = 1, 2, \ldots, m)$ について $f(\boldsymbol{r}_i) = y_i$ となるような関数 f を算出する．データがノイズを含んでいる場合は正則化（VII 部）を利用して $f(\boldsymbol{r}_i) \approx y_i$ となるような関数 f を求める場合もある．

第 33 章

カーネル法

本章では，カーネルを用いて補間を行うカーネル法についてまとめる．

関数 $f : X \to \mathbb{C}$ をサンプル点 $(\boldsymbol{r}_i, y_i) \in X \times \mathbb{C}$ $(i = 1, 2, \ldots, m)$ から補間することを考える．カーネルを用いた補間では，カーネル $K : X \times X \to \mathbb{C}$ を用いて

$$f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r}, \boldsymbol{r}_i) \tag{33.1}$$

のようにおき，$y_i = f(\boldsymbol{r}_i)$ となるように係数 c_i を決める [60].

また，関数 $p_i : X \to \mathbb{R}$ による項を追加した

$$f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r}, \boldsymbol{r}_i) + \sum_{i=1}^{M} d_i p_i(\boldsymbol{r}) \tag{33.2}$$

を考えて係数 c_i, d_i を求める方法も存在する（33.4 節にて説明する）．

33.1 Tikhonov 正則化による導出

まず，カーネルによる補間の導出を行う．

関数 f はある正規直交基底の存在する関数空間 $\mathcal{H}$ に属するものとし，その関数空間 $\mathcal{H}$ には正規直交基底となる関数 $\alpha_1(\boldsymbol{r}), \alpha_2(\boldsymbol{r}), \ldots, \alpha_N(\boldsymbol{r})$ が存在するものとする[*1]．それらの基底を用いて関数 f を次のようにおく．

$$f(\boldsymbol{r}) = \sum_{i=1}^{N} x_i \alpha_i(\boldsymbol{r}) \tag{33.3}$$

ここで，$A_{ij} = \alpha_j(\boldsymbol{r}_i)$ となる行列 $A \in \mathbb{C}^{m \times N}$ を考えると，最小二乗法の評価関数は

$$\|A\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{33.4}$$

となる．

これに Tikhonov 正則化（28 章）を適用する．関数空間 $\mathcal{H}$ のノルムを $\|\cdot\|_{\mathcal{H}}$ とすると，α_i が $\mathcal{H}$ の正規直交基底であることから，

$$\begin{aligned} \|f\|_{\mathcal{H}}^2 &= \sum_{i=1}^{N} |x_i|^2 \\ &= \|\boldsymbol{x}\|_2^2 \end{aligned} \tag{33.5}$$

[*1] この導出では N が有限であるとする．

となる．よって，Tikhonov 正則化の評価関数は式 (28.1) のように書ける．

式 (28.9) より最適解は

$$\boldsymbol{x} = A^*(AA^* + \lambda I)^{-1}\boldsymbol{y} \tag{33.6}$$

となる．ここで，行列 AA^* の (i,j) 要素は

$$[AA^*]_{ij} = \sum_{k=1}^{N} \alpha_k(\boldsymbol{r}_i)\overline{\alpha_k(\boldsymbol{r}_j)} \tag{33.7}$$

となる．それをもとに次のように関数 $K : X \times X \to \mathbb{C}$ を定義する．

$$K(\boldsymbol{r}, \boldsymbol{r}') = \sum_{k=1}^{N} \alpha_k(\boldsymbol{r})\overline{\alpha_k(\boldsymbol{r}')} \tag{33.8}$$

この関数がカーネルである．また，行列 $K_m \in \mathbb{C}^{m\times m}$ を $K_m = AA^*$ のようにおく．なお，この行列は半正定値エルミート行列である．

ここで，変数 $\boldsymbol{c} \in \mathbb{C}^m$ を

$$\boldsymbol{c} = (K_m + \lambda I)^{-1}\boldsymbol{y} \tag{33.9}$$

のようにおく．このとき，$\boldsymbol{x} = A^*\boldsymbol{c}$ だから，

$$\begin{aligned} f(\boldsymbol{r}) &= \sum_{i=1}^{N} x_i \alpha_i(\boldsymbol{r}) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{m} \overline{A_{ji}} c_j \alpha_i(\boldsymbol{r}) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{m} \overline{\alpha_i(\boldsymbol{r}_j)} c_j \alpha_i(\boldsymbol{r}) \\ &= \sum_{j=1}^{m} c_j K(\boldsymbol{r}, \boldsymbol{r}_j) \end{aligned} \tag{33.10}$$

となる．

関数 f を正規直交基底で表していたことから，カーネル法は関数空間 $\mathcal{H}$ 全体の中で Tikhonov 正則化の評価関数

$$\sum_{i=1}^{m} |y_i - f(\boldsymbol{r}_i)|^2 + \|f\|_{\mathcal{H}}^2 \tag{33.11}$$

を最小化したものを求められるということが分かる．また，ここでは導出において関数空間 $\mathcal{H}$ の次元を表す N が有限であることを前提としていたが，導出されたカーネル法で用いる行列は $m \times m$ の正方行列であるため，$N \to \infty$ の場合にも適用できる．

33.2　Gaussian Process

カーネルによる補間は，Gaussian Process と呼ばれる確率分布から導くこともできる [61]．ただし，データとカーネル関数の値は実数とする．

平均 $\mu(\boldsymbol{r})$，分散 $K(\boldsymbol{r}, \boldsymbol{r}')$ の Gaussian Process に従う関数 f は，関数値のベクトル $(f(\boldsymbol{r}_1), f(\boldsymbol{r}_2), \ldots, f(\boldsymbol{r}_m))$ が正規分布 $\mathcal{N}(\boldsymbol{\mu}_m, K_m)$ に従う．ここで，$\boldsymbol{\mu}_m$ は $[\boldsymbol{\mu}_m]_i = \mu(\boldsymbol{r}_i)$ なるベクトルであり，K_m は $[K_m]_{ij} = K(\boldsymbol{r}_i, \boldsymbol{r}_j)$ なる行列である．Gaussian Process は関数における正規分布のようなものである．

関数 f が平均 0，分散 $\tau K(\boldsymbol{r},\boldsymbol{r}')$ の Gaussian Process に従っており，関数 f に対するノイズ入りのサンプルが $y_i = f(\boldsymbol{r}_i) + \varepsilon_i$ のように得られているとする．ただし，ε_i は独立に正規分布 $\mathcal{N}(0,\sigma^2)$ に従うものとする．また，$\tau > 0$ は分散の大きさを示すパラメータである．このとき，ベクトル $(y_1, y_2, \ldots, y_m, f(\boldsymbol{r}))$ は次のような正規分布に従う．

$$\mathcal{N}\left(\boldsymbol{0}, \begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix}\right) \tag{33.12}$$

このとき，確率密度関数 $p(y_1, y_2, \ldots, y_m, f(\boldsymbol{r}))$ は次のように書ける．

$$p(y_1, y_2, \ldots, y_m, f(\boldsymbol{r})) = \frac{1}{\sqrt{(2\pi)^{m+1} \det \begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix}}} \exp\left(-\frac{1}{2} \begin{pmatrix} \boldsymbol{y} \\ f(\boldsymbol{r}) \end{pmatrix}^T \begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{y} \\ f(\boldsymbol{r}) \end{pmatrix}\right) \tag{33.13}$$

ここで，

$$\sigma_K^2(\boldsymbol{r}) \equiv \tau K(\boldsymbol{r},\boldsymbol{r}) - \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \tau\boldsymbol{k}(\boldsymbol{r}) \tag{33.14}$$

$$= \tau K(\boldsymbol{r},\boldsymbol{r}) - \tau\boldsymbol{k}(\boldsymbol{r})^T (K_m + \sigma^2 I)^{-1} \boldsymbol{k}(\boldsymbol{r}) \tag{33.15}$$

とおくと，

$$\det \begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix} = \sigma_K^2(\boldsymbol{r}) \det(\tau K_m + \sigma^2 I) \tag{33.16}$$

となる．また，

$$\begin{aligned} &\begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix}^{-1} \\ &= \begin{pmatrix} (\tau K_m + \sigma^2 I)^{-1} + (\tau K_m + \sigma^2 I)^{-1} \tau\boldsymbol{k}(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} & -(\tau K_m + \sigma^2 I)^{-1} \tau\boldsymbol{k}(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} \\ -\sigma_K^2(\boldsymbol{r})^{-1} \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} & \sigma_K^2(\boldsymbol{r})^{-1} \end{pmatrix} \end{aligned} \tag{33.17}$$

であり，

$$\begin{aligned} &\begin{pmatrix} \boldsymbol{y} \\ f(\boldsymbol{r}) \end{pmatrix}^T \begin{pmatrix} \tau K_m + \sigma^2 I & \tau\boldsymbol{k}(\boldsymbol{r}) \\ \tau\boldsymbol{k}(\boldsymbol{r})^T & \tau K(\boldsymbol{r},\boldsymbol{r}) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{y} \\ f(\boldsymbol{r}) \end{pmatrix} \\ &= \boldsymbol{y}^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} + \boldsymbol{y}^T (\tau K_m + \sigma^2 I)^{-1} \tau\boldsymbol{k}(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} \\ &\quad - f(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} - \boldsymbol{y}^T (\tau K_m + \sigma^2 I)^{-1} \tau\boldsymbol{k}(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} f(\boldsymbol{r}) \\ &\quad + f(\boldsymbol{r}) \sigma_K^2(\boldsymbol{r})^{-1} f(\boldsymbol{r}) \\ &= \boldsymbol{y}^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} + \sigma_K^2(\boldsymbol{r})^{-1} \left(f(\boldsymbol{r}) - \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y}\right)^2 \end{aligned} \tag{33.18}$$

となる．よって，確率密度関数は次のように分割できる．

$$p(y_1, y_2, \ldots, y_m, f(\boldsymbol{r})) = p_{\boldsymbol{y}}(\boldsymbol{y}) p_f(f(\boldsymbol{r})) \tag{33.19}$$

$$p_{\boldsymbol{y}}(\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^m \det(\tau K_m + \sigma^2 I)}} \exp\left(-\frac{1}{2} \boldsymbol{y}^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y}\right) \tag{33.20}$$

$$p_f(f(\boldsymbol{r})) = \frac{1}{\sqrt{(2\pi)^m \sigma_K^2(\boldsymbol{r})}} \exp\left(-\frac{1}{2} \sigma_K^2(\boldsymbol{r})^{-1} \left(f(\boldsymbol{r}) - \mu_K(\boldsymbol{r})\right)^2\right) \tag{33.21}$$

$$\mu_K(\boldsymbol{r}) = \tau\boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} \tag{33.22}$$

分割したあとの $p_{\boldsymbol{y}}$ は $\boldsymbol{y}$ に関する正規分布 $\mathcal{N}(\boldsymbol{0}, \tau K_m + \sigma^2 I)$ であり，p_f は $f(\boldsymbol{r})$ に関する正規分布 $\mathcal{N}(\mu_K(\boldsymbol{r}), \sigma_K^2(\boldsymbol{r}))$ である．$\lambda = \sigma^2/\tau$ とおくと，

$$\begin{aligned}\mu_K(\boldsymbol{r}) &= \tau \boldsymbol{k}(\boldsymbol{r})^T (\tau K_m + \sigma^2 I)^{-1} \boldsymbol{y} \\ &= \boldsymbol{k}(\boldsymbol{r})^T (K_m + \lambda I)^{-1} \boldsymbol{y} \\ &= \sum_{i=1}^{m} K(\boldsymbol{r}, \boldsymbol{r}_i) \left[(K_m + \lambda I)^{-1} \boldsymbol{y} \right]_i \end{aligned} \tag{33.23}$$

のように前節と同様の公式が得られる．

Gaussian Process を用いた表現では分散 $\sigma_K^2(\boldsymbol{r})$ も得られるため，補間された関数のどの部分に誤差が多い可能性が高いか評価するのにも役立つ．また，パラメータ推定において確率の最大化を行うといった応用もある（33.7 節）．

33.3　RBF 補間

カーネル $K(\boldsymbol{r}, \boldsymbol{r}')$ の値が距離 $\|\boldsymbol{r} - \boldsymbol{r}'\|$ に依存する場合，そのもとになる関数は Radial Basis Function (RBF) と呼ばれる．RBF $\varphi : [0, \infty) \to \mathbb{R}$ によるカーネルは

$$K(\boldsymbol{r}, \boldsymbol{r}') = \varphi\left(\frac{\|\boldsymbol{r} - \boldsymbol{r}'\|}{c} \right) \tag{33.24}$$

のように書ける．RBF としては表 33.1 のようなものが挙げられる [61, 62].

33.3.1　Wendland の Compactly Supported RBF

RBF 補間を行うにあたって式 (33.9) にあるような係数の計算が必要となるが，表 33.1 で挙げたような RBF を使用した場合，係数行列 K_m は $m \times m$ の密行列となるため，補間に用いるサンプル点の数 m の数が増加すると計算に必要な時間が m^3 オーダーで増加し，メモリは m^2 オーダーで増加する．そこで，係数行列 K_m が疎行列となるような RBF が考案された．そのような RBF のうち，ここでは Wendland の compactly supported RBF [63] について説明する．

Wendland の compactly supported RBF においては，

$$x_+^l = \begin{cases} x^l & \text{for x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{33.25}$$

のような truncated power function を用いる．また，サンプル点の座標は d 次元の Euclid 空間にあるものとする．

表 33.1: RBF の例 [61, 62]

名称	関数
Gaussian	e^{-r^2}
Multi-quadric	$\sqrt{1 + r^2}$
Inverse Multi-quadric	$1/\sqrt{1 + r^2}$
Inverse Quadric	$1/(1 + r^2)$
Sech	$1/\cosh r$
Bessel ($d = 1, 2, \ldots$)	$J_{d/2-1}(r)/r^{d/2-1}$
Compactly Supported RBF	（33.3.1 節参照）

まず，truncated power function を用いて定義される連続な RBF

$$\varphi_l(r) \equiv (1-r)_+^l \tag{33.26}$$

のパラメータ l を $l \leq \lfloor d/2 \rfloor + 1$ となるように設定すると，カーネルの行列 K_m が正定値になる．さらに，微分も可能な RBF を作るため，積分の作用素

$$I(f)(r) \equiv \int_r^\infty s f(s) ds \tag{33.27}$$

を定義し，その作用素を φ_l に適用した RBF

$$\varphi_{l,k} \equiv I^k \varphi_l \tag{33.28}$$

を考える．このとき，$l = \lfloor d/2 \rfloor + k + 1$ とすると，カーネルの行列 K_m が正定値になる特性を持ったまま C^{2k} 級の関数になる [63, Theorem 3.5]．積分の作用素を適用した結果は部分積分を用いて計算でき，$k = 0, 1, 2$ における RBF は以下のような式で書ける（導出過程は後述する）．

$$\varphi_{l,0} = (1-r)_+^l \tag{33.29}$$

$$\varphi_{l,1} = \frac{1}{(l+1)(l+2)} (1-r)_+^{l+1} \left((l+1)r + 1 \right) \tag{33.30}$$

$$\varphi_{l,2} = \frac{1}{(l+1)(l+2)(l+3)(l+4)} (1-r)_+^{l+2} \left((l+1)(l+3)r^2 + 3(l+2)r + 3 \right) \tag{33.31}$$

これらをプロットすると図 33.1 のようになる．

なお，微分の計算をするにあたっては，積分の作用素 I の逆となる微分の作用素が

$$D(f)(r) \equiv -\frac{1}{r} \frac{d}{dr} f(r) \tag{33.32}$$

で定義されるということを利用すればよい．

Wendland の CSRBF の計算式の導出

ここで，式 (33.30), (33.31) の導出過程について説明しておく．

まず，式 (33.30) を導出する．式 (33.28) の定義より，

$$\begin{aligned} \varphi_{l,1}(r) &= I(\varphi_l)(r) \\ &= \int_r^\infty s \varphi_l(s) ds \\ &= \int_r^\infty s (1-s)_+^l ds \end{aligned} \tag{33.33}$$

と書ける．ここで，truncated power function の定義より，被積分関数は $s \geq 1$ において 0 となるため，$r \geq 1$ において $\varphi_{l,1} = 0$ となる．そこで，$0 \leq r < 1$ の場合について計算を続行する．部分積分を用いることで，

$$\begin{aligned} \varphi_{l,1}(r) &= \int_r^1 s(1-s)^l ds \\ &= \left[s \left(-\frac{1}{l+1} \right) (1-s)^{l+1} \right]_r^1 - \int_r^1 \left(-\frac{1}{l+1} \right) (1-s)^{l+1} ds \\ &= \frac{1}{l+1} r (1-r)^{l+1} - \left[\frac{1}{(l+1)(l+2)} (1-s)^{l+2} \right]_r^1 \\ &= \frac{1}{l+1} r (1-r)^{l+1} + \frac{1}{(l+1)(l+2)} (1-r)^{l+2} \end{aligned} \tag{33.34}$$

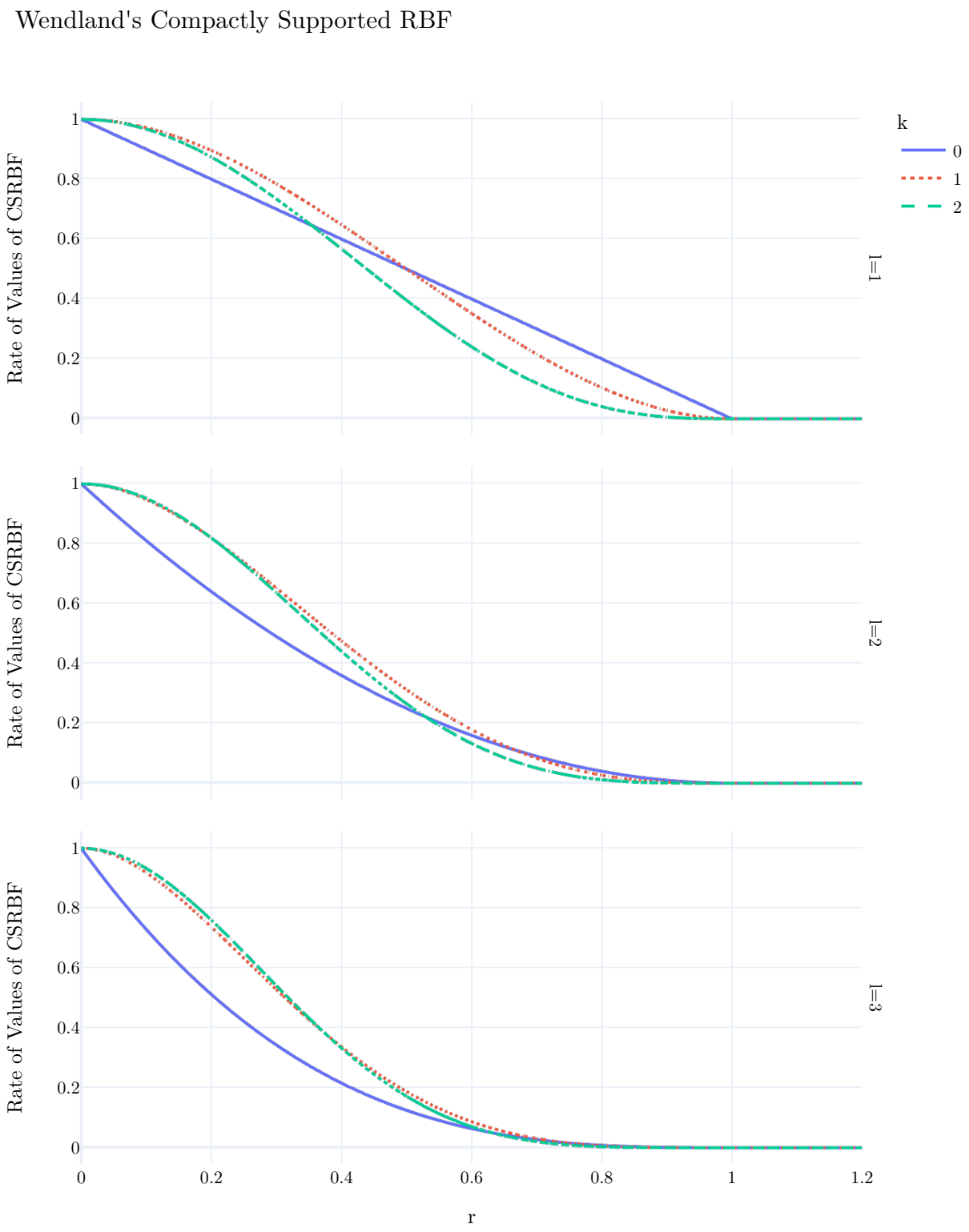


図 33.1: Wendland の Compactly Supported RBF $\varphi_{l,k}(r)$ [63] の例（見やすいように $r = 0$ のときの値に対する比をプロットしている．）

となる．式を整理すると，

$$\begin{aligned}\varphi_{l,1}(r) &= \frac{1}{(l+1)(l+2)}(1-r)^{l+1}\left((l+2)r+(1-r)\right)\\ &= \frac{1}{(l+1)(l+2)}(1-r)^{l+1}\left((l+1)r+1\right)\end{aligned} \tag{33.35}$$

となる．最後に $r \geq 1$ で $\varphi_{l,1} = 0$ となるように truncated power function を用いて書き直すことで，

$$\varphi_{l,1}(r) = \frac{1}{(l+1)(l+2)}(1-r)_+^{l+1}\left((l+1)r+1\right) \tag{33.36}$$

が得られる．

続いて，式 (33.31) を導出する．$\varphi_{l,1}$ と同様に定義より

$$\begin{aligned}\varphi_{l,2} &= I(\varphi_{l,1})(r)\\ &= \int_r^\infty s\varphi_{l,1}(s)ds\\ &= \int_r^\infty s\frac{1}{(l+1)(l+2)}(1-s)_+^{l+1}\left((l+1)s+1\right)ds\\ &= \frac{1}{(l+1)(l+2)}\int_r^\infty s(1-s)_+^{l+1}\left((l+1)s+1\right)ds\end{aligned} \tag{33.37}$$

となる．$r \geq 1$ においては $\varphi_{l,2} = 0$ となるため，$0 \leq r < 1$ とすると，

$$\varphi_{l,2} = \frac{1}{(l+1)(l+2)}\int_r^1 s(1-s)^{l+1}\left((l+1)s+1\right)ds \tag{33.38}$$

となる．部分積分を 2 回適用し，式を整理すると，

$$\begin{aligned}\varphi_{l,2} &= \frac{1}{(l+1)(l+2)}\int_r^1\left((l+1)s^2+s\right)(1-s)^{l+1}ds\\ &= \frac{1}{(l+1)(l+2)}\left[\left((l+1)s^2+s\right)\left(-\frac{1}{l+2}\right)(1-s)^{l+2}\right]_r^1\\ &\qquad -\frac{1}{(l+1)(l+2)}\int_r^1(2(l+1)s+1)\left(-\frac{1}{l+2}\right)(1-s)^{l+2}ds\\ &= \frac{1}{(l+1)(l+2)^2}\left((l+1)r^2+r\right)(1-r)^{l+2}\\ &\qquad -\frac{1}{(l+1)(l+2)}\left[(2(l+1)s+1)\frac{1}{(l+2)(l+3)}(1-s)^{l+3}\right]_r^1\\ &\qquad +\frac{1}{(l+1)(l+2)}\int_r^1 2(l+1)\frac{1}{(l+2)(l+3)}(1-s)^{l+3}ds\\ &= \frac{1}{(l+1)(l+2)^2}\left((l+1)r^2+r\right)(1-r)^{l+2}\\ &\qquad +\frac{1}{(l+1)(l+2)^2(l+3)}\left(2(l+1)r+1\right)(1-r)^{l+3}\\ &\qquad +\frac{1}{(l+1)(l+2)}\left[-\frac{2(l+1)}{(l+2)(l+3)(l+4)}(1-s)^{l+4}\right]_r^1\\ &= \frac{1}{(l+1)(l+2)^2}\left((l+1)r^2+r\right)(1-r)^{l+2}\\ &\qquad +\frac{1}{(l+1)(l+2)^2(l+3)}\left(2(l+1)r+1\right)(1-r)^{l+3}\\ &\qquad +\frac{2}{(l+2)^2(l+3)(l+4)}(1-r)^{l+4}\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{(l+1)(l+2)^2(l+3)(l+4)}(1-r)^{l+2} \\
&\qquad \Big((l+3)(l+4)\Big((l+1)r^2+r\Big) + (l+4)\,(2(l+1)r+1)\,(1-r) + 2(l+1)(1-r)^2\Big) \\
&= \frac{1}{(l+1)(l+2)^2(l+3)(l+4)}(1-r)^{l+2} \\
&\qquad \big((l+1)(l+3)(l+4)r^2 + (l+3)(l+4)r \\
&\qquad - 2(l+1)(l+4)r^2 + (2l+1)(l+4)r + (l+4) \\
&\qquad + 2(l+1)r^2 - 4(l+1)r + 2(l+1)\big) \\
&= \frac{1}{(l+1)(l+2)^2(l+3)(l+4)}(1-r)^{l+2}\Big((l+1)(l+2)(l+3)r^2 + 3(l+2)^2 r + 3(l+2)\Big) \\
&= \frac{1}{(l+1)(l+2)(l+3)(l+4)}(1-r)^{l+2}\Big((l+1)(l+3)r^2 + 3(l+2)r + 3\Big) \qquad (33.39)
\end{aligned}
$$

となる．最後に $r \geq 1$ で $\varphi_{l,2} = 0$ となるように truncated power function を用いて書き直すことで，

$$
\varphi_{l,2} = \frac{1}{(l+1)(l+2)(l+3)(l+4)}(1-r)_+^{l+2}\Big((l+1)(l+3)r^2 + 3(l+2)r + 3\Big) \qquad (33.40)
$$

が得られる．

33.4　多項式などによる項の追加

本章の冒頭で概説したように，カーネル法で

$$
f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r}, \boldsymbol{r}_i) + \sum_{i=1}^{M} d_i p_i(\boldsymbol{r}) \qquad (33.41)
$$

のように追加の関数 $p_i : X \to \mathbb{R}$ による項を追加することが考えられる．

例えば，RBF 補間においては追加の項が定数や多項式となるように，$M = 1$ で $p_1(\boldsymbol{r}) = 1$ としたり，$\boldsymbol{r} = (x, y)^\top$ のときに $M = 3$ で $p_1(\boldsymbol{r}) = 1$, $p_2(\boldsymbol{r}) = x$, $p_3(\boldsymbol{r}) = y$ としたりする．これにより，補間の精度が高くなるという [62, Section 3.1.3.5]．また，thin plate spline [64], spherical spline [65] のように，追加の項が指定されているカーネルも存在する．

このような補間においては，係数 c_i, d_i を以下の方程式により決定する．

$$
\begin{pmatrix} K_m + \lambda I & P \\ P^\top & O \end{pmatrix}\begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} \qquad (33.42)
$$

ここで，行列 $P \in \mathbb{R}^{m \times M}$ の成分は $P_{ij} = p_j(\boldsymbol{r}_i)$ とする．

33.5　再生核ヒルベルト空間

本節では，再生核ヒルベルト空間 (Reproducing Kernel Hilbert Space, RKHS) について説明するとともに，再生核ヒルベルト空間とカーネル法との関連について説明する．

再生核ヒルベルト空間は次のように定義される．

定義 33.1 ([66])**.** 空間 X 上で定義される関数のヒルベルト空間 $\mathcal{H}$ があり，関数 $f, g \in \mathcal{H}$ の内積は (f, g) で表されるものとする．このとき，関数 $K : X \times X \to \mathbb{C}$ が以下を満たすのであれば，関数 K を再生核と呼び，$\mathcal{H}$ は再生核ヒルベルト空間と呼ぶ．

- 任意の $y \in X$ について y を固定した x についての関数 $K(x, y)$ は $\mathcal{H}$ に属する．

- 任意の $y \in X$ と $f \in \mathcal{X}$ について，

$$f(y) = (f(x), K(x,y))_x \tag{33.43}$$

が成り立つ．（添え字 x は x の関数として内積をとることを示す．）

上記の定義を満たすような再生核 K をカーネル法におけるカーネルとして使用する場合，点群 $\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m$ とカーネルによる行列

$$K_m = \begin{pmatrix} K(\boldsymbol{r}_1,\boldsymbol{r}_1) & K(\boldsymbol{r}_1,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_1,\boldsymbol{r}_m) \\ K(\boldsymbol{r}_2,\boldsymbol{r}_1) & K(\boldsymbol{r}_2,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_2,\boldsymbol{r}_m) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{r}_m,\boldsymbol{r}_1) & K(\boldsymbol{r}_m,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_m,\boldsymbol{r}_m) \end{pmatrix} \tag{33.44}$$

は少なくとも半正定値になる [66]．点群を適切に選ぶことで行列 K_m が正則になるようにした場合，カーネル法による補間は以下の性質を満たすことが示せる[*2]．

定理 33.1. 空間 X 上で定義される関数の再生核ヒルベルト空間 $\mathcal{H}$ があり，関数 $f, g \in \mathcal{H}$ の内積は (f,g) で表されるものとし，ノルムを $\|f\|$ とし，再生核は $K : X \times X \to \mathbb{C}$ とする．また，$i = 1, 2, \ldots, m$ について点 $\boldsymbol{r}_i \in X$ と値 $f_i \in C$ を定義する．このとき，$f(\boldsymbol{r}_i) = f_i$ を満たす $f \in \mathcal{H}$ でノルム $\|f\|$ が最小となるものは以下で与えられる．

$$f(\boldsymbol{r}) = \sum_{j=1}^{m} c_j K(\boldsymbol{r}, \boldsymbol{r}_j) \tag{33.45}$$

ここで，c_j は以下を満たす係数である．

$$\begin{pmatrix} K(\boldsymbol{r}_1,\boldsymbol{r}_1) & K(\boldsymbol{r}_1,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_1,\boldsymbol{r}_m) \\ K(\boldsymbol{r}_2,\boldsymbol{r}_1) & K(\boldsymbol{r}_2,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_2,\boldsymbol{r}_m) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{r}_m,\boldsymbol{r}_1) & K(\boldsymbol{r}_m,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_m,\boldsymbol{r}_m) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{pmatrix} \tag{33.46}$$

証明．式 (33.46) を満たす係数 c_i について，

$$f(\boldsymbol{r}) = \sum_{j=1}^{m} c_j K(\boldsymbol{r}, \boldsymbol{r}_j) + g(\boldsymbol{r}) \tag{33.47}$$

となる関数 $g \in \mathcal{H}$ を考える．両辺について右から $K(\boldsymbol{r}, \boldsymbol{r}_i)$ との内積をとると，

$$(f(\boldsymbol{r}), K(\boldsymbol{r}, \boldsymbol{r}_i))_{\boldsymbol{r}} = \sum_{j=1}^{m} c_j (K(\boldsymbol{r}, \boldsymbol{r}_j), K(\boldsymbol{r}, \boldsymbol{r}_i))_{\boldsymbol{r}} + (g(\boldsymbol{r}), K(\boldsymbol{r}, \boldsymbol{r}_i))_{\boldsymbol{r}} \tag{33.48}$$

となる．再生核の定義を用いると以下のように変形できる．

$$\begin{aligned} f(\boldsymbol{r}_i) &= \sum_{j=1}^{m} c_j K(\boldsymbol{r}_i, \boldsymbol{r}_j) + (g(\boldsymbol{r}), K(\boldsymbol{r}, \boldsymbol{r}_i)) \\ f_i &= f_i + (g(\boldsymbol{r}), K(\boldsymbol{r}, \boldsymbol{r}_i)) \\ (g(\boldsymbol{r}), K(\boldsymbol{r}, \boldsymbol{r}_i))_{\boldsymbol{r}} &= 0 \end{aligned} \tag{33.49}$$

この結果を利用すると，関数 f のノルムは

$$\|f\|^2 = (f, f)$$

[*2] 文献 [65,67] の理論を利用している．

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i\bar{c_j}(K(\boldsymbol{r},\boldsymbol{r}_i),K(\boldsymbol{r},\boldsymbol{r}_j)) + \sum_{j=1}^{m}\bar{c_j}(g(\boldsymbol{r}),K(\boldsymbol{r},\boldsymbol{r}_j)) + \sum_{j=1}^{m} c_j(K(\boldsymbol{r},\boldsymbol{r}_j),g(\boldsymbol{r})) + \|g\|^2$$
$$= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i\bar{c_j}(K(\boldsymbol{r},\boldsymbol{r}_i),K(\boldsymbol{r},\boldsymbol{r}_j)) + \|g\|^2 \tag{33.50}$$

となる．これが最小となる $g \in \mathcal{H}$ は $\|g\| = 0$ となる $g(\boldsymbol{r}) = 0$ である．よって，$g(\boldsymbol{r}) = 0$ とした

$$f(\boldsymbol{r}) = \sum_{j=1}^{m} c_j K(\boldsymbol{r},\boldsymbol{r}_j) \tag{33.51}$$

は $f(\boldsymbol{r}_i) = f_i$ となる $f \in \mathcal{H}$ の中で $\|f\|$ が最も小さいものとなっている． □

定理 33.2. 空間 X 上で定義される関数のヒルベルト空間 $\mathcal{H}$ があり，関数 $f, g \in \mathcal{H}$ の内積は (f, g) で表されるものとし，ノルムを $\|f\|$ とする．空間 $\mathcal{H}$ は部分空間 $\mathcal{H}_0$ と $\mathcal{H}_1$ の直積で表されるものとし，空間 $\mathcal{H}$ に属する関数を部分空間 $\mathcal{H}_0$, $\mathcal{H}_1$ へ射影する演算子 P_0, P_1 を定義する．部分空間 $\mathcal{H}_0$ は再生核 $K : X \times X \to \mathbb{C}$ による再生核ヒルベルト空間とし，部分空間 $\mathcal{H}_1$ に属する関数は基底 $p_i in \mathcal{H}_1$ $(i = 1, 2, \ldots, M)$ により表現できるものとする．また，$i = 1, 2, \ldots, m$ について点 $\boldsymbol{r}_i \in X$ と値 $f_i \in C$ を定義する．ただし，行列

$$K_m = \begin{pmatrix} K(\boldsymbol{r}_1,\boldsymbol{r}_1) & K(\boldsymbol{r}_1,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_1,\boldsymbol{r}_m) \\ K(\boldsymbol{r}_2,\boldsymbol{r}_1) & K(\boldsymbol{r}_2,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_2,\boldsymbol{r}_m) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{r}_m,\boldsymbol{r}_1) & K(\boldsymbol{r}_m,\boldsymbol{r}_2) & \cdots & K(\boldsymbol{r}_m,\boldsymbol{r}_m) \end{pmatrix} \tag{33.52}$$

は正則であるものとし，行列

$$P = \begin{pmatrix} p_1(\boldsymbol{r}_1) & p_2(\boldsymbol{r}_1) & \cdots & p_M(\boldsymbol{r}_1) \\ p_1(\boldsymbol{r}_2) & p_2(\boldsymbol{r}_2) & \cdots & p_M(\boldsymbol{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\boldsymbol{r}_m) & p_2(\boldsymbol{r}_m) & \cdots & p_M(\boldsymbol{r}_m) \end{pmatrix} \tag{33.53}$$

は列フルランクとする．このとき，$f(\boldsymbol{r}_i) = f_i$ を満たす $f \in \mathcal{H}$ でノルム $\|P_0 f\|$ が最小となるものは以下で与えられる．

$$f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r},\boldsymbol{r}_i) + \sum_{i=1}^{M} d_i p_i(\boldsymbol{r}) \tag{33.54}$$

ここで，c_i, d_i は以下を満たす係数である．

$$\begin{pmatrix} K_m & P \\ P^* & O \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{pmatrix} \tag{33.55}$$

証明．まず，

$$f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r},\boldsymbol{r}_i) + \sum_{i=1}^{M} d_i p_i(\boldsymbol{r}) \tag{33.56}$$

とおいた場合を考える．$f(\boldsymbol{r}_i) = f_i$ より，

$$\sum_{j=1}^{m} c_j K(\boldsymbol{r}_i,\boldsymbol{r}_j) + \sum_{j=1}^{M} d_j p_j(\boldsymbol{r}_i) = f_i \tag{33.57}$$

となる．これを行列で表すと

$$K_m \boldsymbol{c} + P\boldsymbol{d} = \boldsymbol{f} \tag{33.58}$$

となる．また，関数 f の空間 $\mathcal{H}_0$ におけるノルムは

$$\begin{aligned}\|P_0 f\|^2 &= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i \bar{c_j}(K(\boldsymbol{r},\boldsymbol{r}_i), K(\boldsymbol{r},\boldsymbol{r}_j))_{\boldsymbol{r}} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i \bar{c_j} K(\boldsymbol{r}_j,\boldsymbol{r}_i) \\ &= \boldsymbol{c}^* K \boldsymbol{c} \end{aligned} \tag{33.59}$$

と書ける．ここで，K_m は正則であることから，

$$\boldsymbol{c} = K_m^{-1}(\boldsymbol{f} - P\boldsymbol{d}) \tag{33.60}$$

となる．また，再生核の性質 $K(\boldsymbol{x},\boldsymbol{y}) = \overline{K(\boldsymbol{y},\boldsymbol{x})}$ より K_m はエルミート行列になるため，

$$\boldsymbol{c}^* = (\boldsymbol{f} - P\boldsymbol{d})^* K_m^{-1} \tag{33.61}$$

が成り立つ．よって，

$$\begin{aligned}\|P_0 f\|^2 &= \boldsymbol{c}^* K \boldsymbol{c} \\ &= (\boldsymbol{f} - P\boldsymbol{d})^* K_m^{-1} K K_m^{-1}(\boldsymbol{f} - P\boldsymbol{d}) \\ &= (\boldsymbol{f} - P\boldsymbol{d})^* K_m^{-1}(\boldsymbol{f} - P\boldsymbol{d}) \\ &= \boldsymbol{f}^* K_m^{-1}\boldsymbol{f} - \boldsymbol{f}^* K_m^{-1} P\boldsymbol{d} - \boldsymbol{d}^* P^* K_m^{-1}\boldsymbol{f} + \boldsymbol{d}^* P^* K_m^{-1} P\boldsymbol{d} \end{aligned} \tag{33.62}$$

となる．ここで，再生核の性質と K_m が正則であることから，K_m は正定値の行列となる．さらに，P が列フルランクであることから，$P^* K_m^{-1} P$ も正定値の行列となる．このことを利用すると，

$$\begin{aligned}\|P_0 f\|^2 = \left(\boldsymbol{d} - \left(P^* K_m^{-1} P\right)^{-1} P^* K_m^{-1}\boldsymbol{f}\right)^* P^* K_m^{-1} P \left(\boldsymbol{d} - \left(P^* K_m^{-1} P\right)^{-1} P^* K_m^{-1}\boldsymbol{f}\right) \\ - \boldsymbol{f}^* K_m^{-1} P^* \left(P^* K_m^{-1} P\right)^{-1} P^* K_m^{-1}\boldsymbol{f} + \boldsymbol{f}^* K_m^{-1}\boldsymbol{f} \end{aligned} \tag{33.63}$$

と変形できる．$P^* K_m^{-1} P$ が正定値であることから，$\|P_0 f\|^2$ を最小化する $\boldsymbol{d}$ は第一項を 0 にする

$$\boldsymbol{d} = \left(P^* K_m^{-1} P\right)^{-1} P^* K_m^{-1}\boldsymbol{f} \tag{33.64}$$

である．このとき，

$$\begin{aligned}\boldsymbol{c} &= K_m^{-1}(\boldsymbol{f} - P\boldsymbol{d}) \\ &= K_m^{-1}\left(I - P\left(P^* K_m^{-1} P\right)^{-1} P^* K_m^{-1}\right)\boldsymbol{f} \end{aligned} \tag{33.65}$$

となる．これらの $\boldsymbol{c}$ と $\boldsymbol{d}$ は方程式 (33.55) の解になっている．

定理 33.1 と同様にすると，

$$f(\boldsymbol{r}) = \sum_{i=1}^{m} c_i K(\boldsymbol{r},\boldsymbol{r}_i) + \sum_{i=1}^{M} d_i p_i(\boldsymbol{r}) \tag{33.66}$$

が $f(\boldsymbol{r}_i) = f_i$ となる $f \in \mathcal{H}$ の中で $\|P_0 f\|$ を最小化することも示せる． □

これらの定理の結果を利用し，例えばノルムを小さくすることで関数が滑らかになるように再生核ヒルベルト空間 $\mathcal{H}$ を定義すると，カーネル法により与えられた点を通る最も滑らかな関数を得ることができる．実際に，そのような理論で作られたカーネルとして thin plate spline [64], spherical spline [65] が挙げられる．

33.5.1 Thin plate spline

Thin plate spline [64] は，$\mathbb{R}^d$ $(d = 1, 2, \ldots)$ から $\mathbb{R}$ への関数を n 回偏微分したものの積を積分するようなノルム

$$(f, g) = \sum_{\alpha_1 + \cdots + \alpha_d = n} \frac{n!}{\alpha_1! \cdots \alpha_d!} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \frac{\partial^n f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \frac{\partial^n g}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} dx_1 \cdots dx_d \tag{33.67}$$

により定義される関数空間を考える．この内積で定義される関数空間は $2n - d > 0$ の場合に再生核

$$K(\boldsymbol{r}, \boldsymbol{s}) = \begin{cases} \dfrac{(-1)^{d/2+1+n}}{2^{2n} \pi^{d/2} (n-1)! (n - d/2)!} \|\boldsymbol{r} - \boldsymbol{s}\|_2^{2n-d} \log \|\boldsymbol{r} - \boldsymbol{s}\|_2^2 & 2n - d \text{ が偶数の場合} \\ \dfrac{\Gamma(d/2 - n)}{2^{2n} \pi^{d/2} (n-1)!} \|\boldsymbol{r} - \boldsymbol{s}\|_2^{2n-d} & 2n - d \text{ が奇数の場合} \end{cases} \tag{33.68}$$

を持つ再生核ヒルベルト空間となる [64]．ただし，内積の中の微分に $n - 1$ 次までの多項式が含まれないため，d 個の変数の $n - 1$ 次までの組み合わせを関数 $p_1, p_2, \ldots, p_M$ とし，定理 33.2 を適用して補間を行う．

33.5.2 Spherical spline

Spherical spline [65] は，単位球面 S から実数への関数について n 回微分して積分するノルム

$$\|f\|^2 = \begin{cases} \displaystyle\int_0^{2\pi} \int_0^{\pi} \left(\triangle^{n/2} f \right)^2 \sin\theta d\theta d\varphi & n \text{ が偶数の場合} \\ \displaystyle\int_0^{2\pi} \int_0^{\pi} \left(\frac{1}{\sin^2\theta} \left(\frac{\partial \triangle^{(n-1)/2} f}{\partial \varphi} \right)^2 + \left(\frac{\partial \triangle^{(n-1)/2} f}{\partial \theta} \right)^2 \right) \sin\theta d\theta d\varphi & n \text{ が奇数の場合} \end{cases} \tag{33.69}$$

により定義される関数空間を考える（θ と φ は極座標における緯度と経度を示す）．$n > 1$ の場合，この関数空間は再生核

$$K(\boldsymbol{r}, \boldsymbol{s}) \equiv \frac{1}{4\pi} \sum_{v=1}^{\infty} \frac{2v + 1}{v^n (v+1)^n} P_v(\cos\gamma(\boldsymbol{r}, \boldsymbol{s})) \tag{33.70}$$

を持つ．ここで，$P_v(x)$ は v 次の Legendre 関数（3.1 節）で，$\gamma(\boldsymbol{r}, \boldsymbol{s})$ は $\boldsymbol{r}$ と $\boldsymbol{s}$ をベクトルとしたときに 2 つのベクトルがなす角を示す．この再生核に加えて，関数 $p_1(\boldsymbol{r}) = 1$ を用いて定理 33.2 のように補間を行う．

ここで，再生核の値の計算に無限級数の評価が必要となるが，$n = 2$ の場合は

$$K(\boldsymbol{r}, \boldsymbol{s}) = \frac{1}{4\pi} \int_0^1 \log h \left(1 - \frac{1}{h} \right) \left(\frac{1}{\sqrt{1 - 2h\cos\gamma(\boldsymbol{r}, \boldsymbol{s}) + h^2}} - 1 \right) dh \tag{33.71}$$

と書くことができ [65]，比較的容易な数値積分により再生核の値を計算できる．

また，文献 [65] 内では比較的計算が容易なカーネル

$$K'(\boldsymbol{r}, \boldsymbol{s}) \equiv \frac{1}{2\pi} \sum_{v=1}^{\infty} \frac{1}{(v+1)(v+2)\cdots(v+2n-1)} P_v(\cos\gamma(\boldsymbol{r}, \boldsymbol{s})) \tag{33.72}$$

$$= \frac{1}{2\pi} \left(\frac{1}{(2n-2)!} \int_0^1 \frac{(1-h)^{2n-2}}{\sqrt{1 - 2h\cos\gamma(\boldsymbol{r}, \boldsymbol{s}) + h^2}} dh - \frac{1}{(2n-1)!} \right) \tag{33.73}$$

も提案されている．

33.6 固有値分解による数値解法

密行列の固有値分解を用いてカーネル法の計算を行うことを考える．

33.6.1 追加の項がない場合

まず，本節では 33.4 節のような追加の項がない場合の方程式 $K_m\boldsymbol{c} = \boldsymbol{y}$ を考える．カーネルの値を並べた行列 $K_m \in \mathbb{C}^{m\times m}$ を次のように固有値分解する．

$$K_m = VDV^* \tag{33.74}$$

ここで，$V \in \mathbb{C}^{m\times m}$ はエルミート行列であり，$D \in \mathbb{R}^{m\times m}$ は実数による対角行列である．ここで，K_m は半正定値とする（このような性質を持つカーネルは正定値カーネルと呼ばれる [60]）．33.1 節で導出したカーネルや 33.3 節で示した Gaussian の RBF によるカーネルは正定値カーネルである．このとき，

$$K_m + \lambda I = V(D + \lambda I)V^* \tag{33.75}$$

が成り立つ．$\lambda > 0$ であれば確実に $K_m + \lambda I$ が正定値となり，逆行列を持つ．よって，式 (33.9) で定めた係数 $\boldsymbol{c}$ は

$$\begin{aligned}\boldsymbol{c} &= (K_m + \lambda I)^{-1}\boldsymbol{y} \\ &= V(D + \lambda I)^{-1}V^*\boldsymbol{y}\end{aligned} \tag{33.76}$$

と書ける．さらに，$V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m)$, $D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ とおくと，

$$\boldsymbol{c} = \sum_{i=1}^{m} \frac{1}{\lambda_i + \lambda}(\boldsymbol{v}_i^*\boldsymbol{y})\boldsymbol{v}_i \tag{33.77}$$

のように表すこともできる．

33.6.2 追加の項がある場合

続いて，本節では 33.4 節のような追加の項がある場合の方程式

$$\begin{pmatrix} K_m + \lambda I & P \\ P^\top & O \end{pmatrix}\begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} \tag{33.78}$$

を考える．ただし，サンプル点と追加の項の関数 p_i を適切に選択することにより行列 P のランクが M となるようにしておく．また，行列 K_m は実数による行列で正定値とする．

追加で出てきた行列 P について以下のように QR 分解を行う．

$$P = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} R \\ O \end{pmatrix} \tag{33.79}$$

ここで，$Q_1 \in \mathbb{R}^{m\times M}$, $Q_2 \in \mathbb{R}^{m\times(m-M)}$ $R \in \mathbb{R}^{M\times M}$ とする．また，行列 P のランクが M としているため，行列 R は正則行列となる．

方程式のうち $P^\top\boldsymbol{c} = \boldsymbol{0}$ より，

$$\begin{aligned}\boldsymbol{c}^\top P &= \boldsymbol{0} \\ \boldsymbol{c}^\top Q_1 R &= \boldsymbol{0} \\ \boldsymbol{c}^\top Q_1 &= \boldsymbol{0}\end{aligned} \tag{33.80}$$

となるため，$\boldsymbol{c}$ は Q_1 のすべての列に対して直行するように選択する必要がある．QR 分解において (Q_1, Q_2) は直行行列になることから，$\boldsymbol{c}$ は Q_2 の列ベクトルにより作られる部分空間に属している必要があるため，$\boldsymbol{c} = Q_2\boldsymbol{\gamma}$ のようにベクトル $\boldsymbol{\gamma}$ を用いて書ける．

方程式の残りの $(K_m + \lambda I)\boldsymbol{c} + P\boldsymbol{d} = \boldsymbol{y}$ について，変数を置き換えると

$$(K_m + \lambda I)Q_2\boldsymbol{\gamma} + Q_1R\boldsymbol{d} = \boldsymbol{y} \tag{33.81}$$

となる．左から $Q_2^\top$ をかけると，$Q_2^\top Q_1 = O$ より

$$Q_2^\top(K_m + \lambda I)Q_2\boldsymbol{\gamma} = Q_2^\top\boldsymbol{y} \tag{33.82}$$

となる．$(K_m + \lambda I)$ が正定値となることから，フルランクの行列 Q_2 をかけた $Q_2^\top(K_m + \lambda I)Q_2$ も正定値（つまり正則でもある）となり，

$$\boldsymbol{\gamma} = \left(Q_2^\top(K_m + \lambda I)Q_2\right)^{-1} Q_2^\top\boldsymbol{y} \tag{33.83}$$

となる．よって，$\boldsymbol{c} = Q_2\boldsymbol{\gamma}$ より

$$\boldsymbol{c} = Q_2\left(Q_2^\top(K_m + \lambda I)Q_2\right)^{-1} Q_2^\top\boldsymbol{y} \tag{33.84}$$

となる．

また，$(K_m + \lambda I)\boldsymbol{c} + P\boldsymbol{d} = \boldsymbol{y}$ より

$$\begin{aligned}
P\boldsymbol{d} &= \boldsymbol{y} - (K_m + \lambda I)\boldsymbol{c} \\
Q_1R\boldsymbol{d} &= \boldsymbol{y} - (K_m + \lambda I)\boldsymbol{c} \\
R\boldsymbol{d} &= Q_1^\top\left(\boldsymbol{y} - (K_m + \lambda I)\boldsymbol{c}\right) \\
\boldsymbol{d} &= R^{-1}Q_1^\top\left(\boldsymbol{y} - (K_m + \lambda I)\boldsymbol{c}\right) \\
\boldsymbol{d} &= R^{-1}Q_1^\top\left(\boldsymbol{y} - K_m\boldsymbol{c}\right)
\end{aligned} \tag{33.85}$$

となる．Moore-Penrose の一般化逆行列を用いて $\boldsymbol{d} = R^\dagger\left(\boldsymbol{y} - K_m\boldsymbol{c}\right)$ としても良い[*3]．

さらに，$Q_2^\top K_mQ_2$ の固有値分解 $Q_2^\top K_mQ_2 = UDU^\top$（$D$ は対角行列で U は直行行列）が得られている場合，

$$\begin{aligned}
\boldsymbol{c} &= Q_2\left(Q_2^\top(K_m + \lambda I)Q_2\right)^{-1} Q_2^\top\boldsymbol{y} \\
&= Q_2\left(Q_2^\top K_mQ_2 + \lambda I\right)^{-1} Q_2^\top\boldsymbol{y} \\
&= Q_2\left(UDU^\top + \lambda I\right)^{-1} Q_2^\top\boldsymbol{y} \\
&= Q_2U\left(D + \lambda I\right)^{-1} U^\top Q_2^\top\boldsymbol{y} \\
&= Q_2U\left(D + \lambda I\right)^{-1} (Q_2U)^\top\boldsymbol{y}
\end{aligned} \tag{33.86}$$

のように計算できる．

33.7　パラメータ推定

式 (33.24) の定数 c のようにカーネルがパラメータを持っている場合がある．そのようなパラメータの推定を [68, Remark 3 (Connection to spatial statistics)] に沿った maximum likelihood estimation (MLE) により考える．なお，本節ではカーネルの値やデータの値がすべて実数になっているものとする．

[*3] 処理系が QR 分解の結果を用いて方程式を解くようなプログラムを用意している場合がある．

Gaussian Process（33.2 節）より，データ $\boldsymbol{y}$ の確率密度関数は

$$p(\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^m \det(\tau K_m + \sigma^2 I)}} \exp\left(-\frac{1}{2}\boldsymbol{y}^T(\tau K_m + \sigma^2 I)^{-1}\boldsymbol{y}\right) \tag{33.87}$$

であり，その対数を取ると

$$\log p(\boldsymbol{y}) = -\frac{m}{2}\log(2\pi) - \frac{1}{2}\log\det(\tau K_m + \sigma^2 I) - \frac{1}{2}\boldsymbol{y}^T(\tau K_m + \sigma^2 I)^{-1}\boldsymbol{y} \tag{33.88}$$

となる．これを最大化するようにパラメータを決定する．

$\lambda = \sigma^2/\tau$ を固定すると，

$$\log p(\boldsymbol{y}) = -\frac{m}{2}\log(2\pi) - \frac{m}{2}\log\tau - \frac{1}{2}\log\det(K_m + \lambda I) - \frac{1}{2\tau}\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y} \tag{33.89}$$

となる．これを最大化するように τ を決定すると，

$$\tau = \frac{1}{m}\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y} \tag{33.90}$$

となり，そのときの $\log p(\boldsymbol{y})$ は

$$\log p(\boldsymbol{y}) = -\frac{m}{2}\log(2\pi) + \frac{m}{2}\log m - \frac{m}{2}\log(\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y}) - \frac{1}{2}\log\det(K_m + \lambda I) - \frac{m}{2} \tag{33.91}$$

となる．m はサンプルの数であり，定数だから，パラメータ推定では

$$m\log(\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y}) + \log\det(K_m + \lambda I) \tag{33.92}$$

を最小化すれば良い．

なお，33.6 節における固有値分解を用いることで，パラメータ推定における式 (33.90) の τ は

$$\begin{aligned}\tau &= \frac{1}{m}\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y} \\ &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\lambda_i + \lambda}(\boldsymbol{v}_i^T\boldsymbol{y})^2\end{aligned} \tag{33.93}$$

のように書ける．さらに，評価関数は

$$\begin{aligned}&m\log(\boldsymbol{y}^T(K_m + \lambda I)^{-1}\boldsymbol{y}) + \log\det(K_m + \lambda I) \\ &= m\log\left(\sum_{i=1}^{m}\frac{1}{\lambda_i + \lambda}(\boldsymbol{v}_i^T\boldsymbol{y})^2\right) + \log\left(\prod_{i=1}^{m}(\lambda_i + \lambda)\right) \\ &= m\log\left(\sum_{i=1}^{m}\frac{1}{\lambda_i + \lambda}(\boldsymbol{v}_i^T\boldsymbol{y})^2\right) + \sum_{i=1}^{m}\log(\lambda_i + \lambda)\end{aligned} \tag{33.94}$$

のように書ける．

33.8　大域最適解への応用

Gaussian Process（33.2 節）を利用して大域最適解を行う Gaussian Process 最適化が考案されている [69]．

領域 $D \in \mathbb{R}^d$ 上で関数 $f : D \to \mathbb{R}$ を最小化する制約なし最適化の問題を考える[*4]．文献 [69] のアルゴリズムにおいては，各反復ごとにこれまでに算出したサンプル点 $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ $(i = 1, 2, \ldots, T)$ に対して Gaussian Process による補

[*4] 文献 [69] では最大化を前提としているが，ここでは II 部に合わせて最小化を考える．

間を行い，Gaussian Process の平均 $\mu_T(\boldsymbol{x})$ と標準偏差 $\sigma_T(\boldsymbol{x})$ を算出できるようにする．そして，反復ごとに変化する係数 β_t とパラメータ ν を用いて定義される目的関数

$$F_T(\boldsymbol{x}) = \mu_T(\boldsymbol{x}) - \sqrt{\nu\beta_{T+1}}\sigma_T(\boldsymbol{x}) \tag{33.95}$$

を最小化するような $\boldsymbol{x}$ を次のサンプル点として選択する．係数 ν, β_t は

- $\nu = 1/5$, $\beta_t = 2\log(|D|t^2\pi^2/6\delta)$, $\delta \in (0,1)$ [69] [*5]
- $\nu = 1$, $\beta_t = 2\log(t^{d/2+2}\pi^2/3\delta)$, $\delta \in (0,1)$ [61]

のように与えられる．関数 F_T を最小化するために別の大域最適解のアルゴリズムが必要となるが，元の目的関数 f の値を得るのに時間がかかる場合には有効なアルゴリズムである．

[*5] 文献 [69] 内の定理では $\nu = 1$ の場合を扱っていたが，数値実験において $\nu = 1/5$ とすると結果が良くなったという記述がある．また，文献 [69] の数値実験では $\delta = 0.1$ が選択されている．

第 IX 部

付録

付録 A

Lagrangian と Hamiltonian

本章ではラグランジアン（Lagrangian）とハミルトニアン（Hamiltonian）について，[2, Chapter 3] 本書の説明に必要な最低限の理論をまとめておく．

座標 $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)^\top \in \mathbb{R}^d$ とその速度 $\dot{\boldsymbol{q}}$ を用いて位置エネルギーを以下のように表す．

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} a_{ij}(\boldsymbol{q}) \dot{q}_i \dot{q}_j \tag{A.1}$$

これをもとにモーメント $\boldsymbol{p}$ を以下のように定める．

$$\boldsymbol{p} = \frac{\partial T}{\partial \dot{\boldsymbol{q}}} \tag{A.2}$$

さらに，位置エネルギー V を $\boldsymbol{q}$ の式で表しておき，その勾配 $\partial V / \partial q_i$ を求める．このとき，運動方程式は以下のように書ける．（ラグランジュの運動方程式）

$$\dot{p}_i = \frac{\partial T}{\partial q_i} - \frac{\partial V}{\partial q_i} \tag{A.3}$$

ラグランジアン $L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T - V$ を定義すると，以下のようにも書ける．

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} = \frac{\partial L}{\partial q_i} \tag{A.4}$$

また，全エネルギーの和を $\boldsymbol{q}$ と $\boldsymbol{p}$ で表現した関数 $H(\boldsymbol{q}, \boldsymbol{p}) = T + V$ をハミルトニアンと呼ぶ．ハミルトニアンを用いると，運動方程式は次の式で表現される．（正準方程式）

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}} \tag{A.5}$$

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}} \tag{A.6}$$

付録 B

二重振り子

本章では，図 B.1 のように 1 つの振り子の先端に別の振り子を取り付けたような二重振り子について運動方程式を導出する．

xy 平面の原点に固定された回転軸 O に棒 1 の端を取り付け，もう一方の端に回転軸 J_{12} を取り付け，その回転軸に別の棒 2 の端を取り付けた状態になっているとする．各棒は密度と太さが均一で，棒 i（$i = 1, 2$）について

- 質量は m_i
- 長さは L_i
- 重心位置は C_i
- y 軸の負の方向から時計回りの角度は θ_i

とする．この条件で，ラグランジュの運動方程式（式 (A.3)）を考える．

各棒は剛体とみなすと，棒 i の慣性モーメント I_i は以下のように求められる．

$$I_i = \int_{-\frac{L_i}{2}}^{\frac{L_i}{2}} \frac{m_i}{L_i} s^2 ds = \frac{1}{12} m_i L_i^2 \tag{B.1}$$

また，各棒の重心位置は

$$C_1 = \frac{L_1}{2} \begin{pmatrix} \sin\theta_1 \\ -\cos\theta_1 \end{pmatrix} \tag{B.2}$$

$$C_2 = L_1 \begin{pmatrix} \sin\theta_1 \\ -\cos\theta_1 \end{pmatrix} + \frac{L_2}{2} \begin{pmatrix} \sin(\theta_2) \\ -\cos(\theta_2) \end{pmatrix} \tag{B.3}$$

と書ける．それらの速度は

$$\dot{C}_1 = \frac{L_1}{2} \dot{\theta}_1 \begin{pmatrix} \cos\theta_1 \\ \sin\theta_1 \end{pmatrix} \tag{B.4}$$

$$\dot{C}_2 = L_1 \dot{\theta}_1 \begin{pmatrix} \cos\theta_1 \\ \sin\theta_1 \end{pmatrix} + \frac{L_2}{2} \dot{\theta}_2 \begin{pmatrix} \cos\theta_2 \\ \sin\theta_2 \end{pmatrix} \tag{B.5}$$

となり，その大きさの 2 乗は

$$\|\dot{C}_1\|_2^2 = \frac{1}{4} L_1^2 \dot{\theta}_1^2 \tag{B.6}$$

$$\|\dot{C}_2\|_2^2 = L_1^2 \dot{\theta}_1^2 + L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + \frac{1}{4} L_2^2 \dot{\theta}_2^2 \tag{B.7}$$

となる．

これらを用いて，二重振り子の系全体の運動エネルギー T は以下のように書ける．

$$T = \frac{1}{2} m_1 \|\dot{C}_1\|_2^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} m_2 \|\dot{C}_2\|_2^2 + \frac{1}{2} I_2 \dot{\theta}_2^2 \tag{B.8}$$

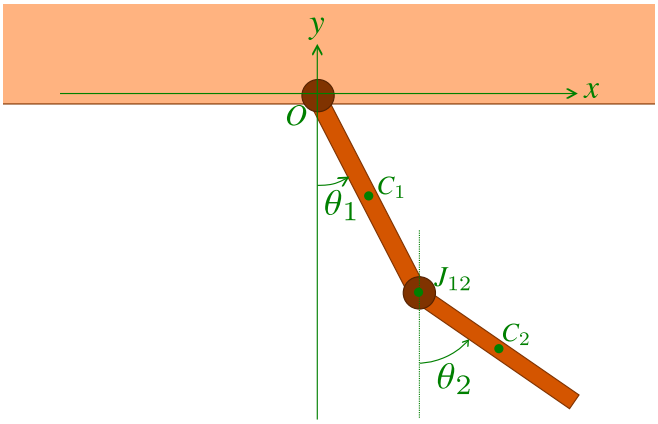


図 B.1: 二重振り子

$$= \frac{1}{8}m_1L_1^2\dot{\theta}_1^2 + \frac{1}{24}m_1L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) + \frac{1}{8}m_2L_2^2\dot{\theta}_2^2 + \frac{1}{24}m_2L_2^2\dot{\theta}_2^2 \tag{B.9}$$

$$= \frac{1}{6}\left(m_1+3m_2\right)L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) + \frac{1}{6}m_2L_2^2\dot{\theta}_2^2 \tag{B.10}$$

位置エネルギー V は重心の y 座標を用いて以下のように書ける．

$$V = -\frac{1}{2}m_1L_1g\cos\theta_1 - m_2\left(L_1g\cos\theta_1 + \frac{1}{2}L_2g\cos\theta_2\right) \tag{B.11}$$

$$= -\frac{1}{2}\left(m_1+2m_2\right)L_1g\cos\theta_1 - \frac{1}{2}m_2L_2g\cos\theta_2 \tag{B.12}$$

上記を用いてモーメントは以下のように求められる．

$$p_1 = \frac{\partial T}{\partial\dot{\theta}_1} \tag{B.13}$$

$$= \frac{1}{3}\left(m_1+3m_2\right)L_1^2\dot{\theta}_1 + \frac{1}{2}m_2L_1L_2\dot{\theta}_2\cos(\theta_1-\theta_2) \tag{B.14}$$

$$p_2 = \frac{\partial T}{\partial\dot{\theta}_2} \tag{B.15}$$

$$= \frac{1}{2}m_2L_1L_2\dot{\theta}_1\cos(\theta_1-\theta_2) + \frac{1}{3}m_2L_2^2\dot{\theta}_2 \tag{B.16}$$

運動方程式は以下のようになる．

$$\dot{p}_1 = \frac{\partial T}{\partial\theta_1} - \frac{\partial V}{\partial\theta_1} \tag{B.17}$$

$$= -\frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2) - \frac{1}{2}\left(m_1+2m_2\right)L_1g\sin\theta_1 \tag{B.18}$$

$$\dot{p}_2 = \frac{\partial T}{\partial\theta_2} - \frac{\partial V}{\partial\theta_2} \tag{B.19}$$

$$= \frac{1}{2}m_2L_1L_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2) - \frac{1}{2}m_2L_2g\sin\theta_2 \tag{B.20}$$

Todo list

対称行列の CG 法と PCG 法に触れておきたい. 18
有限要素法の説明をして，導出過程を書くようにしたい. 22
他のアルゴリズムも実装して比較する. 22

参考文献

[1] Kenta Kabashima. numerical-collection-cpp. https://gitlab.com/MusicScience37Projects/numerical-analysis/numerical-collection-cpp, 2021.

[2] Philip M. Morse and Herman Feshbach. *Methods of Theoretical Physics*. McGraw-Hill Book Company, Inc., 1953.

[3] C. Radhakrishna Rao and Sujit Kumar Mitra. *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, inc., 1971.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 4th edition, 2013.

[5] J. W. Ruge and K. Stüben. *4. Algebraic Multigrid*, chapter 4, pp. 73–130. Society for Industrial and Applied Mathematics, 1987.

[6] C. H. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger. A parallel algebraic multigrid solver for finite element method based source localization in the human brain. *Computing and Visualization in Science*, Vol. 5, pp. 1401–1426, 2002.

[7] Peter Knabner and Lutz Angermann. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*. Springer-Verlag, 2003.

[8] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[9] 森正武, 室田一雄, 杉原正顯. 数値計算の基礎. 株式会社岩波書店, 1993.

[10] D. P. Laurie. Calculation of gauss-kronrod quadrature rules. *Mathematics of Computation*, Vol. 66, No. 219, pp. 1133 – 1145, 1997.

[11] G. H. Golub and J. H. Welsch. Calculation of gauss quadrature rules. *Mathematics of Computation*, Vol. 23, pp. 221 – 230, 1969.

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes : The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[13] Ronald Cools and Philip Rabinowitz. Monomial cubature rules since "stroud": a compilation. *Journal of Computational and Applied Mathematics*, Vol. 48, pp. 309 – 326, 1993.

[14] D. P. Laurie. Cubtri: Automatic cubature over a triangle. *ACM Transaction on Mathematical Software*, Vol. 8, No. 2, pp. 210 – 218, 1982.

[15] David G. Luenberger. *Linear and Nonlinear Programming*. Kluwer Academic Publishers, 2nd edition, 2003.

[16] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[17] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, Vol. 79, No. 1, 1993.

[18] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the direct algorithm. *Journal of Global Optimization*, Vol. 21, pp. 27 – 37, 2001.

[19] Ratko Grbić, Emmanuel Karlo Nyarko, and Rudolf Scitovski. A modification of the direct method for lipschitz global optimization for a symmetric function. *Journal of Global Optimization*, Vol. 57, pp. 1193 – 1212, 2013.

[20] Yaroslav D. Sergeyev. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization*, Vol. 16, No. 3, pp. 910 – 937, 2006.

[21] Yaroslav D. Sergeyev. Efficient strategy for adaptive partition of n-dimensional internals in the framework of diagonal algorithms. *Journal of Optimization Theory and Applications*, Vol. 107, No. 1, pp. 145 – 168, 2000.

[22] Dmitri E. Kvasov and Yaroslav D. Sergeyev. Deterministic approaches for solving practical black-box global optimization problems. *Advances in Engineering Software*, Vol. 80, pp. 58 – 66, 2015. Civil-Comp.

[23] Anna Molinaro, Clara Pizzuti, and Yaroslav D. Sergeyev. Acceleration tools for diagonal information global optimization algorithms. *Computational Optimization and Applications*, Vol. 18, pp. 5 – 26, 2001.

[24] Abdel-Rahman Hedar. Global optimization test problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[25] 戸川隼人. 新装版 UNIX ワークステーションによる 科学技術計算ハンドブック. 株式会社 サイエンス社, 第5th 版, 2007.

[26] 三井斌友. 微分方程式の数値解法 I. 株式会社岩波書店, 1993.

[27] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer-Verlag, 1991.

[28] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer-Verlag, 1993.

[29] Kjell Gustafsson. Control Theoretic Techniques for Stepsize Selection in Explicit Runge-Kutta Methods. *ACM Transactions on Mathematical Software*, Vol. 7, No. 4, pp. 533–554, 1991.

[30] David S. Blom, Philipp Birken, Hester Bijl, Fleur Kessels, Andreas Meister, and Alexander H. van Zuijlen. A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows. *Advances in Computational Mathematics*, Vol. 42, pp. 1401–1426, 2016.

[31] David S. Blom, Hester Bijl, Alexander H. van Zuijlen, Philipp Birken, Kess Vuik, and Richard Dwight. Rosenbrock time integration combined with Krylov subspace enrichment for unsteady flow simulations, 2013. http://resolver.tudelft.nl/uuid:0aeae67a-52b5-40ef-a86a-8b21d5cee18b.

[32] John H. Mathews and Kurtis K. Fink. *Numerical Methods Using Matlab.* Prentice-Hall Inc., 4th edition, 2004.

[33] John Bagterp Jorgensen, Morten Rode Kristensen, and Per Grove Thomsen. A Family of ESDIRK Integration Methods. *arXiv*, Vol. 1803.01613v1, pp. 1–22, 2018.

[34] Christopher A. Kennedy and Mark H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, Vol. 44, , 2003.

[35] H. H. Rosenbrock. Some general implicit processes for the numerical solution of differential equations. *The Computer Journal*, Vol. 5, No. 4, pp. 329 – 330, 1963.

[36] J. Rang and L. Angermann. New rosenbrock w-methods of order 3 for partial differential algebraic equations of index 1. *BIT Numerical Mathematics*, Vol. 45, pp. 761 – 787, 2005.

[37] Gerd Steinebach. RODASP, MATLAB Central File Exchange, 2022. Acquired from https://www.mathworks.com/matlabcentral/fileexchange/10354-rodasp at August 7, 2022.

[38] Joachim Rang. Improved traditional Rosenbrock-Wanner methods for stiff ODEs and DAEs. *Journal of Computational and Applied Mathematics*, Vol. 286, , 2015.

[39] G. R. W. Quispel and D. I. McLaren. A new class of energy-preserving numerical integration methods.

Journal of Physics A: Mathematical and Theoretical, Vol. 41, , 2008.

[40] 吉田春夫. ハミルトン力学系のためのシンプレクティック数値積分法, 2013. https://irdb.nii.ac.jp/00947/0001442652.

[41] Etienne Forest and Ronald D. Ruth. Fourth-order symplectic integration. *Physica D: Nonlinear Phenomena*, Vol. 43, No. 1, pp. 105–117, 1990.

[42] J. Candy and W. Rozmus. A symplectic integration algorithm for separable Hamiltonian functions. *Journal of Computational Physics*, Vol. 92, No. 1, pp. 230–256, 1991.

[43] H. Kahan. Further remarks on reducing truncation errors. *Communication of the ACM*, Vol. 8, No. 1, p. 40, 1965.

[44] Hiroshi Hirayama. Double-double Type Quadruple Precision Arithmetic Library for C++ Languages and its Application. *IPSJ SIG Technical Report*, Vol. 2014-HPC-1, No. 22, pp. 1–7, 2014.

[45] Yozo Hida, Sherry Li, and David Bailey. Quad-double arithmetic: Algorithms, implementation, and application, 2001.

[46] 山中脩也, 大石進一. 倍精度に基づく四倍精度四則演算法の誤差とその応用. 数理解析研究所講究録, Vol. 1791, pp. 216–225, 2012.

[47] Kotakemori Hisashi, Fujii Akihiro, Hasegawa Hidehiko, and Nishida Akira. Fast Quad Precision Iterative Solver Library Lis using SSE2. *IPSJ SIG Technical Report*, Vol. 108, pp. 7–12, 2006.

[48] 山中脩也, 大石進一. 倍精度演算に基づく四倍精度除算・平方根計算法. 2012 年ハイパフォーマンスコンピューティングと計算科学シンポジウム, 2012.

[49] 久保田光一, 伊理正夫. アルゴリズムの自動微分と応用. 株式会社 コロナ社, 1998.

[50] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[51] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. Society for Industrial and Applied Mathematics, 1998.

[52] Lars Eldén. A Weighted Pseudoinverse, Generalized Singular Values, and Constrained Least Squares Problems. *BIT Numerical Mathematics*, Vol. 2, pp. 487–502, 1982.

[53] Per Christian Hansen. Regularization Tools: A Matlab Package for Analysis and Solution of Discrete Ill-posed Problems. *Numerical Algorithms*, Vol. 6, pp. 1–35, 1994.

[54] Per Christian Hansen. Analysis of Discrete Ill-Posed Problems by Means of the L-Curve. *SIAM Review*, Vol. 34, No. 4, pp. 561–580, 1992.

[55] J. L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Society for Industrial and Applied Mathematics, 2012.

[56] Per Christian Hansen and Dianne Prost O'Leary. The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems. *SIAM Journal on Scientific Computing*, Vol. 14, No. 6, pp. 1487–1503, 1993.

[57] Grace Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[58] L. N. Deshpande and Didier A. Girard. Fast Computation of Cross-Validated Robust Splines and Other Non-Linear Smoothing Splines. In *Curves and Surfaces*, pp. 143–148. Academic Press, Boston, 1991.

[59] Didier A. Girard. A Fast 'Monte-Carlo Cross-Validation' Procedure for Large Least Squares Problems with Noisy Data. *Numerische Mathematik*, Vol. 56, pp. 1–23, 1989.

[60] 福水健次. カーネル法入門 – 正定値カーネルによるデータ解析 –. 株式会社朝倉書店, 2010.

[61] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv*, Vol. 1012.2599v1, pp. 1–49, 2010.

[62] Bengt Fornberg and Natasha Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*. Society for Industrial and Applied Mathematics, 2015.

[63] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, Vol. 4, pp. 389–396, 1995.

[64] Arpan Ghosh. Efficient thin plate spline interpolation and its application to adaptive optics. Technical report, Eindhoven University of Technology, 2010.

[65] Grace Wahba. Spline interpolation and smoothing on the sphere. *SIAM Journal on Scientific and Statistical Computing*, Vol. 2, No. 1, pp. 5–16, 1981.

[66] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, Vol. 68, No. 3, pp. 337–404, 1950.

[67] George Kimeldorf. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, Vol. 33, No. 1, 1971.

[68] Michael Scheuerer. An alternative procedure for selecting a good value for the parameter c in RBF-interpolation. *Advances in Computational Mathematics*, Vol. 34, pp. 105–126, 2011.

[69] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Settings: No Regret and Experimental Design. *arXiv*, Vol. 0912.3995v4, pp. 1–17, 2010.

索引

A

Adaptive Diagonal Curves 57
Algebric Multigrid 法 18
AMG-CG 法 20
AMG 法 ..... → Algebraic Multigrid 法
Armijo の条件 49
Average vector field method ..... → 平均ベクトル場法

B

Backtracking Line Search 49
barycentric coordinates 37
BFGS 公式 ..... → Broyden-Fletcher-Goldfarb-Shanno 公式
BiCGstab 18
Broyden-Fletcher-Goldfarb-Shanno 公式 52
Butcher-Kuntzmann 公式 91
Butcher 配列 79

C

CG 法 20
Compactly Supported Radial Basis Function 152
CSRBF 152
CUBTRI 37

D

Davidon-Fletcher-Powell 公式 51
DFP 公式 ..... → Davidon-Fletcher-Powell 公式
DIRECT ..... → Dividing Rectangles 法
Dividing Rectangles 法 55
Double Exponential Formula ..... → 二重指数関数型公式
Downhill simplex 法 53

E

ESDIRK 90
Euler 法 89

G

Gaussian Process 150
Gaussian Process 最適化 163
Gauss-Kronrod 積分公式 30
Gauss-Legendre 積分公式 29
Gauss-Seidel 法 16, 20
Gauss 積分公式 29
GCV ..... → Generalized Cross Validation
Generalized Cross Validation 141
GMRES 85
golden section search ..... → 黄金比探索

H

Hamiltonian ..... → ハミルトニアン

I

ill-posed problem 127
influence matrix 141

J

Jacobi 法 16

K

Kahan Summation 111
Krylov 部分空間法 18

L

L-curve 139
Lagrangian ..... → ラグランジアン
leaving-out-one lemma 141
Legendre 関数 13
Lipschitz 連続 55

M

maximum likelihood estimation 162
Milne's device 83
Monte-Carlo Cross-Validation 144
Moore-Penrose の一般化逆行列 15

N

Newton-Raphson 法 69
Newton 法 51, 69

O

OCV ..... → Ordinary Cross Validation
Ordinary Cross Validation 141
ordinary differential equation ..... → 常微分方程式

P

PI 制御 81
Polak-Ribiere 法 52

Q

QR 分解 15

R

Radial Basis Function 152
RBF 補間 152
Reproducing Kernel Hilbert Space 156
RK4 公式 89
RKF45 公式 89
RKHS ..... → Reproducing Kernel Hilbert Space
ROS3w 公式 92
Rosenbrock 法 92
Runge-Kutta 法 79

S

SDIRK 90
Singular Value Decomposition ..... → 特異値分解
Sphierical spline 160
stiffly accurate 85

T

Thin plate spline 160
Tikhonov 正則化 129

い

一般化 Tikhonov 正則化 133
一般化逆行列 ..... → Moore-Penrose の一般化逆行列
一般化特異値分解 134
陰的 Euler 法 92
陰的 Runge-Kutta 法 80

う
埋め込み型の公式 30, 80

お
黄金比探索 47

か
カーネル法 149
硬い系 80

き
逆問題 127
狭義凸関数 45
強凸関数 45
共役勾配法 52
局所最適解 55

け
減速 Newton 法 72
厳密直線探索 49

こ
後退型自動微分 121
古典的 Runge-Kutta 法 89

さ
最急降下法 51
再生核ヒルベルト空間 156
最適化 43
最適解 43
最適値 43

し
次数 80
自動微分 121
準 Newton 法 51
常微分方程式 77
シンプレクティック積分法 97

す
数値積分 27
ステップ幅 49, 72, 79

せ
正準方程式 95, 97, 167
正則化 127
正則化パラメータ 127
制約条件 43
前進型自動微分 121

た
大域最適化 55
大域最適解 55
田中 Formula 90
田中 Formula2 90
段数 79

ち
直線探索 49

て
適応積分 39

と
特異値分解 16
凸関数 45

に
二重指数関数型公式 30

は
敗者復活算法 111
ハミルトニアン 167
半陰的 Runge-Kutta 法 79

へ
平均ベクトル場法 95

ほ
補間 147

も
モーメント 167
目的関数 43

よ
陽的 Runge-Kutta 法 79

ら
ラグランジアン 167
ラグランジュの運動方程式 167

わ
歪対称行列 95