

# 数理工学実験レポート

## 第 3 章（最小二乗法）

学籍番号 1029366161 中塚一瑛

2025 年 11 月 3 日

### 概要

## 目次

1	はじめに	3
2	最小二乗法とその評価方法	3
2.1	原理・方法	3
2.2	課題 1（重回帰）	4
2.3	課題 2（多項式回帰）	5
2.4	課題 3（分散の観測誤差：Cauchy）	7
2.5	課題 4（入力域の制約と設計）	8
3	重み付き最小二乗法	9
3.1	原理と方法	9
3.2	課題 5（2 次元出力：重み付き最小二乗法）	10
3.3	課題 6（2 次元出力・異分散 2 群に対する重み付き最小二乗）	12
4	推定値の合成と逐次最小二乗法	15
4.1	原理と方法	15
4.2	課題 7（推定値の合成の検証：2 分割データの統合）	16
4.3	課題 7（推定値の合成の検証：非線形基底・分割データ）	18
4.4	課題 9（逐次最小二乗法によるシステム同定）	19
4.5	課題 10（非定常時系列の追従：忘却係数付き RLS）	21
5	カルマンフィルタ、カルマン smoother	23
5.1	原理と方法	23
5.2	課題 11（カルマンフィルタ：フォワード）	25
5.3	課題 12（カルマン smoother）	27
6	交互最小二乗法と K-平均法	29

6.1	原理と方法 . . . . .	29
7	アルゴリズム比較	31
8	結論	31
付録 A	使用コード一覧	32
付録 B	補足導出	32

## 1 はじめに

本実験第2回では、最小二乗法の基礎とその実装手法を学ぶことを目的とする。具体的には、観測データからのパラメータ推定、重み付き・逐次最小二乗法、データ分割・推定値の合成を通して、推定精度と計算効率の違いを実験的に比較する。

## 2 最小二乗法とその評価方法

### 2.1 原理・方法

観測モデルを

$$y_i = f(\theta, x_i) + w_i \quad (1)$$

とする。加法雑音  $w_i$  は平均 0、観測ごとに独立、同一分布であり、共分散は有限と仮定する。ここでは  $f(\theta, x) = \phi(x)\theta$  としてパラメータに対して線形とし、最小二乗問題

$$\min_{\theta} \sum_{i=1}^N \|y_i - \phi(x_i)\theta\|^2 \quad (2)$$

を解く。解は

$$\hat{\theta}_N = \left( \sum_{i=1}^N \phi_i^\top \phi_i \right)^{-1} \sum_{i=1}^N \phi_i^\top y_i, \quad \phi_i := \phi(x_i). \quad (3)$$

また雑音分散が未知の場合の推定量および推定誤差共分散は

$$\hat{\sigma}^2 = \frac{1}{N-n} \sum_{i=1}^N \|y_i - \phi_i \hat{\theta}_N\|^2, \quad (4)$$

$$\widehat{\text{Cov}}(\hat{\theta}_N) = \hat{\sigma}^2 \left( \sum_{i=1}^N \phi_i^\top \phi_i \right)^{-1} \quad (5)$$

で与えられる。

当てはまりの評価には決定係数

$$C = \frac{\sum_{i=1}^N \|\phi_i \hat{\theta}_N - \bar{y}\|^2}{\sum_{i=1}^N \|y_i - \bar{y}\|^2}, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i \quad (6)$$

を用いる。以上は資料 3.2 の線形最小二乗および評価に対応する。

本実験では  $\phi(x)$  として設計した特徴量から行列

$$X = \begin{bmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_N)^\top \end{bmatrix} \quad (7)$$

を構成し、観測データを

$$y = [y_1; \dots; y_N] \quad (8)$$

として用いる．パラメータ推定量は

$$\hat{\theta} = (X^\top X)^{-1} X^\top y \quad (9)$$

で与えられる．

残差は

$$r = y - X\hat{\theta} \quad (10)$$

とし，雑音分散推定量は

$$\hat{\sigma}^2 = \frac{\|r\|^2}{N-p} \quad (11)$$

( $p$  は列数) とする．

パラメータ推定の誤差共分散は

$$\hat{\sigma}^2 (X^\top X)^{-1} \quad (12)$$

で与えられる．

決定係数は

$$C = \frac{\|X\hat{\theta} - \bar{y}\mathbf{1}\|^2}{\|y - \bar{y}\mathbf{1}\|^2} \quad (13)$$

と定義する．

以下の R 関数が上記推定を実装している．

```
1 regression_simple <- function(x, y){
2   theta_hat <- solve(t(x) %*% x) %*% t(x) %*% y
3   sigma2_hat <- as.numeric(t(y - x %*% theta_hat) %*%
4     (y - x %*% theta_hat) /
5     (nrow(x) - ncol(x)))
6   err_cov_mat <- sigma2_hat * solve(t(x) %*% x)
7   det_coef <- sum((x %*% theta_hat - mean(y))^2) /
8     sum((y - mean(y))^2)
9   list(theta_hat=theta_hat, err_cov_mat=err_cov_mat,
10     sigma2_hat=sigma2_hat, det_coef=det_coef)
11 }
```

## 2.2 課題 1 (重回帰)

■モデル  $y_i = x_i^\top \theta + w_i$  ( $i = 1, \dots, N$ ).  $x_i \in \mathbb{R}^2$ .  $w_i$  は独立，同分散  $\sigma^2$ ，平均 0.  $X = [x_1^\top; \dots; x_N^\top] \in \mathbb{R}^{N \times 2}$ ,  $y = [y_1; \dots, y_N] \in \mathbb{R}^N$ .

■推定量 最小二乗推定量

$$\hat{\theta}_N = (X^\top X)^{-1} X^\top y.$$

残差  $r = y - X\hat{\theta}_N$  により

$$\hat{\sigma}^2 = \frac{\|r\|_2^2}{N-p}, \quad p=2, \quad \widehat{\text{Cov}}(\hat{\theta}_N) = \hat{\sigma}^2 (X^\top X)^{-1}.$$

決定係数

$$R^2 = \frac{\|X\hat{\theta}_N - \bar{y}\mathbf{1}\|_2^2}{\|y - \bar{y}\mathbf{1}\|_2^2}, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y_i.$$

■実装 データから設計行列  $X$  と観測ベクトル  $y$  を構成し,  $\hat{\theta} = (X^\top X)^{-1} X^\top y$  により推定量を計算する. 残差  $r = y - X\hat{\theta}$  から雑音分散推定量  $\hat{\sigma}^2 = \|r\|^2/(N-p)$  を求め, 推定誤差共分散  $\hat{\sigma}^2(X^\top X)^{-1}$  および決定係数  $R^2$  を算出する関数を実装した.

■収束確認  $N \in \{2, 4, 8, \dots, 2^{13} = 8192\}$  で  $\hat{\theta}_N$  を計算し,  $N$  を横軸とする片対数図で各成分を同一図に描く.

■結果 (全データ  $N = 10000$ )

$$\hat{\theta}_N = \begin{bmatrix} 1.506551 \\ 1.997696 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_N) = \begin{bmatrix} 9.866491 \times 10^{-5} & -4.081657 \times 10^{-7} \\ -4.081657 \times 10^{-7} & 1.005248 \times 10^{-4} \end{bmatrix}.$$

決定係数

$$R^2 = 0.8629734.$$

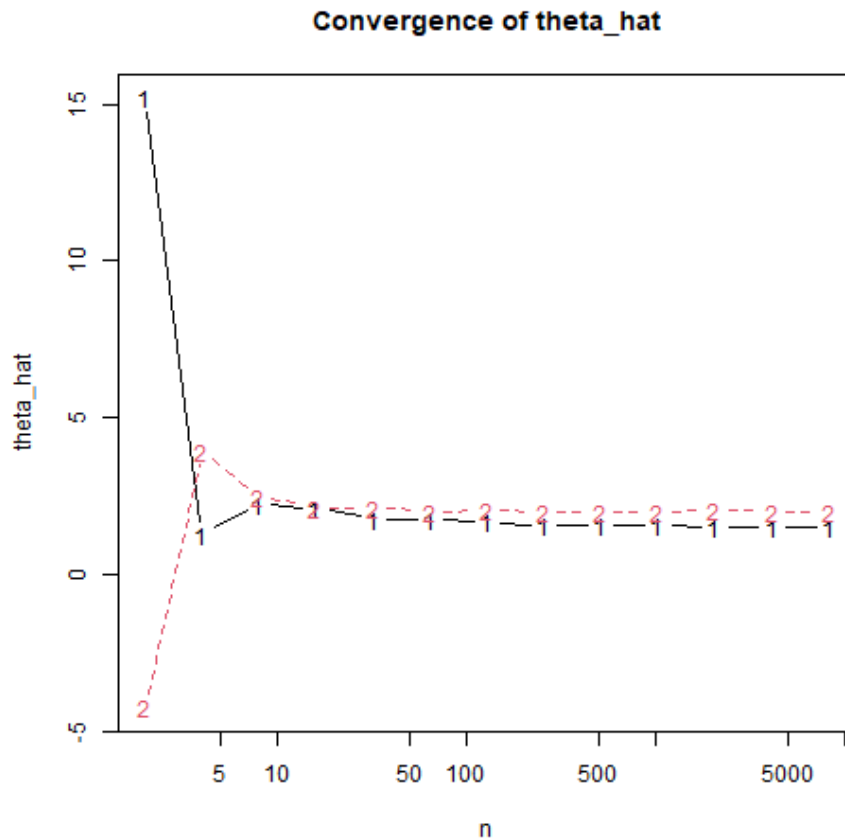


図 1:  $\hat{\theta}_N$  の収束 (横軸  $N = 2, 4, \dots, 8192$  の片対数)

■考察 グラフの結果から,  $\hat{\theta}_N$  は  $N$  を増やすにつれて収束していることが確認できた.

## 2.3 課題 2 (多項式回帰)

■モデル  $y_i = \varphi(x_i)\theta + w_i$ ,  $\varphi(x) = [1 \ x \ x^2 \ x^3]$ ,  $i = 1, \dots, N$ . 雑音  $w_i$  は独立, 同分散  $\sigma^2$ , 平均 0. 分散 9 の正規分布.  $X = [\varphi(x_1); \dots; \varphi(x_N)] \in \mathbb{R}^{N \times 4}$ .

■推定量 推定量・共分散・決定係数は課題 1 と同じ形式 ( $p = 4$ ) で計算する.

■実装 入力  $x_i$  から特徴ベクトル  $\varphi(x_i) = [1 \ x_i \ x_i^2 \ x_i^3]$  を構成し, 設計行列  $X$  を作成する. 課題 1 と同様の関数を用いてパラメータ推定量, 誤差共分散, 決定係数を算出する.  $N \in \{4, 8, 16, \dots, 8192\}$  の各点で推定量を計算し, 片対数プロットで収束の様子を可視化した.

■収束確認  $N \in \{4, 8, 16, \dots, 8192\}$  で  $\hat{\theta}_N$  を計算し,  $N$  を横軸とする片対数図で各成分を同一図に描画する.

■結果 (全データ  $N = 10000$ )

$$\hat{\theta}_N = \begin{bmatrix} -0.50902942 \\ 1.97586067 \\ 0.19774405 \\ -0.09866691 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_N) = \begin{bmatrix} 2.023442 \times 10^{-3} & -1.186497 \times 10^{-5} & -1.348312 \times 10^{-4} & 3.971165 \times 10^{-7} \\ -1.186497 \times 10^{-5} & 6.753015 \times 10^{-4} & -1.898545 \times 10^{-7} & -3.794719 \times 10^{-5} \\ -1.348312 \times 10^{-4} & -1.898545 \times 10^{-7} & 1.603910 \times 10^{-5} & 6.351820 \times 10^{-8} \\ 3.971165 \times 10^{-7} & -3.794719 \times 10^{-5} & 6.351820 \times 10^{-8} & 2.528719 \times 10^{-6} \end{bmatrix}$$

標準誤差 (対角の平方根):  $\text{SE}(\hat{\theta}_0) \approx 0.0450$ ,  $\text{SE}(\hat{\theta}_1) \approx 0.0260$ ,  $\text{SE}(\hat{\theta}_2) \approx 0.00400$ ,  $\text{SE}(\hat{\theta}_3) \approx 0.00159$ .

決定係数:

$$R^2 = 0.461855.$$

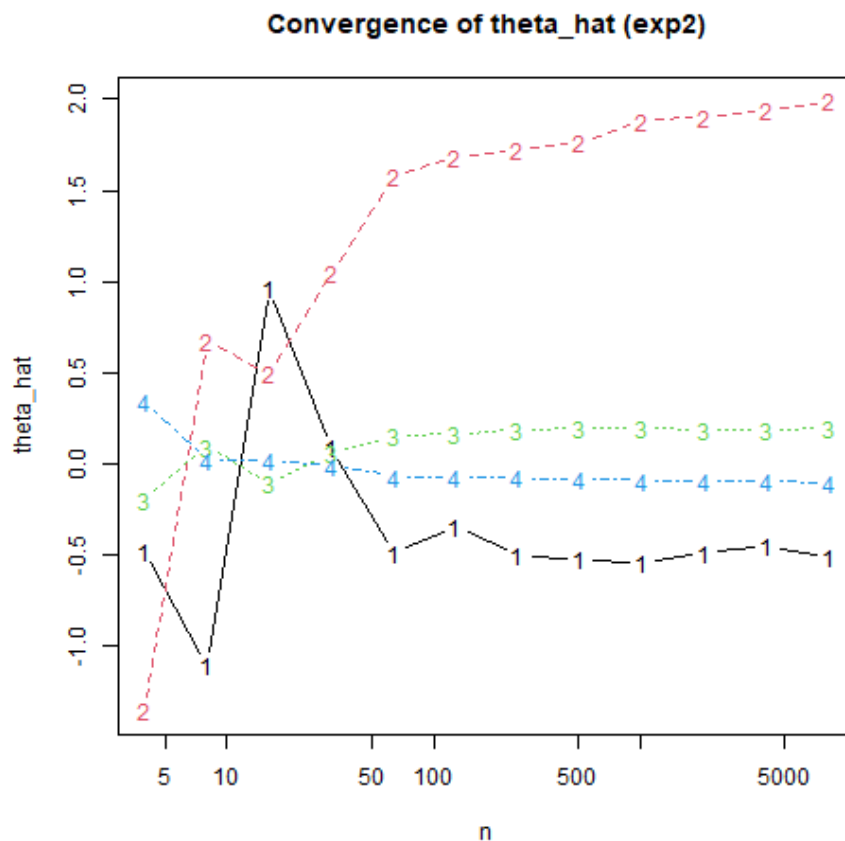


図 2:  $\hat{\theta}_N$  の収束 (横軸  $N = 4, 8, \dots, 8192$  の片対数)

■考察

## 2.4 課題 3 (分散の観測誤差: Cauchy)

■モデルと注意  $y_i = x_i^\top \theta + w_i$  ( $i = 1, \dots, N$ ),  $x_i \in \mathbb{R}^2$ . 観測誤差  $w_i \sim \text{Cauchy}(0, 1)$  独立同分布.

■推定 (形式的な OLS) 推定量・共分散・決定係数は課題 1 と同じ形式 ( $p = 2$ ) で計算する.

■実装 課題 1 と同じ最小二乗推定の枠組みを用いて,  $N \in \{2, 4, \dots, 8192\}$  の各サンプルサイズで推定量を計算した. Cauchy 分布は分散が定義されないため, 通常の推定誤差共分散や決定係数は理論的根拠を持たないが, 形式的に計算することで推定の不安定性を確認する.

■結果 (全データ  $N = 10000$ )

$$\hat{\theta}_N = \begin{bmatrix} 2.373074 \\ 1.537311 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_N) = \begin{bmatrix} 9.234588215 & -0.003642121 \\ -0.003642121 & 9.264075136 \end{bmatrix}.$$

参考: 対角の平方根は  $\text{SE}(\hat{\theta}_1) \approx 3.0388$ ,  $\text{SE}(\hat{\theta}_2) \approx 3.0437$ . 決定係数 (参考値) は

$$R^2 = 0.0002841468.$$

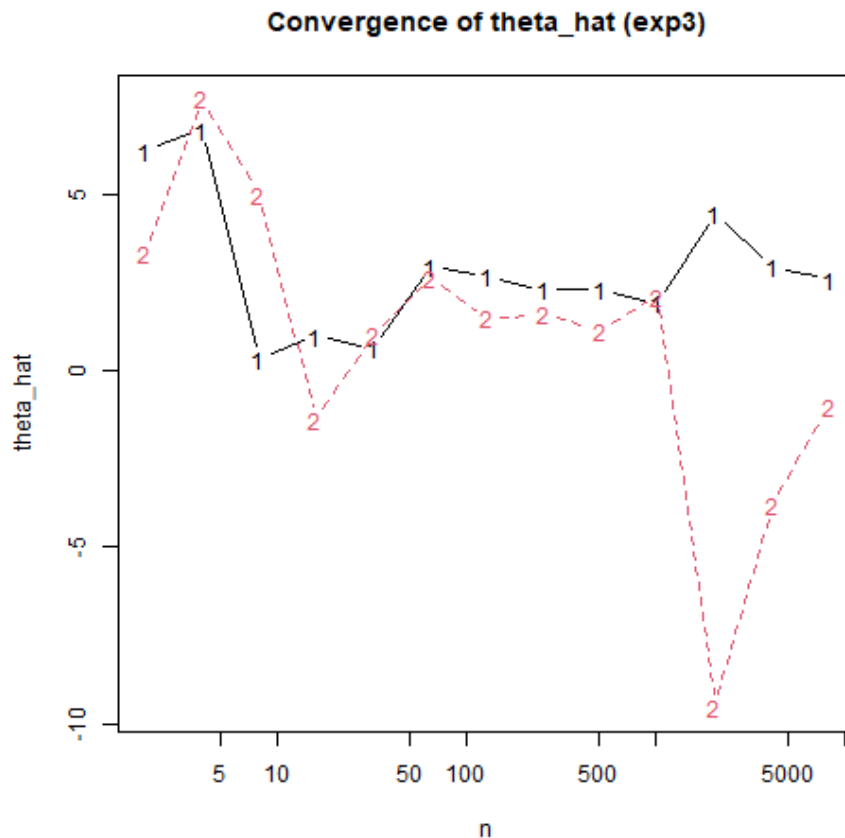


図 3:  $\hat{\theta}_N$  の非収束例 (横軸  $N = 2, 4, \dots, 8192$  の片対数)

■考察 資料にあるように, Cauchy 誤差では外れ値の影響が支配的で,  $\hat{\theta}_N$  は  $N$  を増やしても安定しにくいことが確認できた.

## 2.5 課題 4 (入力域の制約と設計)

■**設定** 課題 2 と同じ  $\varphi(x) = [1 \ x \ x^2 \ x^3]$ , 真の  $\theta$ , 観測誤差  $w_i \sim \mathcal{N}(0, 9)$  とする. ただし入力  $x_i \in [0, 1]$ ,  $i = 1, \dots, 10000$ .  $X = [\varphi(x_1); \dots; \varphi(x_N)] \in \mathbb{R}^{N \times 4}$ ,  $y = [y_1; \dots; y_N]$ .

■**推定量** 推定量・共分散・決定係数は課題 1 と同じ形式 ( $p = 4$ ) で計算する.

■**実装** 課題 2 と同様に  $\varphi(x) = [1 \ x \ x^2 \ x^3]$  の特徴ベクトルを用いて設計行列を構成し, 全データ  $N = 10000$  に対して最小二乗推定を行った.

■**結果** ( $N = 10000$ )

$$\hat{\theta}_{(4)} = \begin{bmatrix} \text{ここに数値} \\ \text{ここに数値} \\ \text{ここに数値} \\ \text{ここに数値} \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_{(4)}) = \begin{bmatrix} \text{ここに } 4 \times 4 \text{ 行列} \end{bmatrix}, \quad R^2 = \begin{bmatrix} \text{ここに数値} \end{bmatrix}.$$

(上の枠に, 直上の R 出力を転記)

■**課題 2.1 との比較** ( $N = 10000$ ) 課題 2.1 の推定結果:

$$\hat{\theta}_{(2.1)} = \begin{bmatrix} -0.50902942 \\ 1.97586067 \\ 0.19774405 \\ -0.09866691 \end{bmatrix}, \quad R^2 = 0.461855.$$

評価観点:

- 分散は  $\text{Var}(\hat{\theta}) = \sigma^2(X^\top X)^{-1}$  に比例.  $x \in [0, 1]$  では列  $\{1, x, x^2, x^3\}$  が強く相関しやすく,  $X^\top X$  の条件が悪化し  $\hat{\theta}$  の不確かさが増えやすい.
- $R^2$  は母集団の  $y$  の散らばり ( $SST$ ) に依存. 入力域が狭いと  $SST$  が小さく, 雑音分散が同じなら  $R^2$  は低下しやすい.

■**考察** (どうデータを取るべきか)

- 目的は  $\text{Var}(\hat{\theta})$  の縮小 (=  $X^\top X$  を「大きく」「良条件」に).
- 入力設計:  $x$  を区間全体で広く配置 (例: 一様), 中心化・標準化して  $[-1, 1]$  に写像, または直交基底 (Legendre/Chebyshev) で回帰.
- 最適化観点: A/D 最適設計を用いて  $\text{tr}((X^\top X)^{-1})$  や  $\det((X^\top X)^{-1})$  を最小化する点集合を選ぶ.



### 3 重み付き最小二乗法

#### 3.1 原理と方法

■原理 各観測  $i = 1, \dots, N$  で  $y_i \in \mathbb{R}^m$ ,  $X_i \in \mathbb{R}^{m \times p}$  とし

$$y_i = X_i \theta + w_i, \quad \mathbb{E}[w_i] = 0, \quad \text{Cov}(w_i) = V \text{ (既知)},$$

を仮定する。 $Q := V^{-1}$  とおく。WLS は重み付き残差平方和

$$J(\theta) = \sum_{i=1}^N (y_i - X_i \theta)^\top Q (y_i - X_i \theta)$$

を最小化する推定で、正規方程式は

$$S \hat{\theta} = b, \quad S := \sum_{i=1}^N X_i^\top Q X_i, \quad b := \sum_{i=1}^N X_i^\top Q y_i.$$

したがって

$$\hat{\theta} = \left( \sum_{i=1}^N X_i^\top Q X_i \right)^{-1} \left( \sum_{i=1}^N X_i^\top Q y_i \right).$$

$V$  が既知のとき

$$\text{Cov}(\hat{\theta}) = S^{-1}.$$

$V = \sigma^2 \Sigma$  のようにスケール未知なら

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^N r_i^\top \Sigma^{-1} r_i}{Nm - p}, \quad r_i := y_i - X_i \hat{\theta}, \quad \widehat{\text{Cov}}(\hat{\theta}) = \hat{\sigma}^2 \left( \sum_{i=1}^N X_i^\top \Sigma^{-1} X_i \right)^{-1}.$$

■方法 (1) 各  $i$  の設計行列  $X_i$  と観測  $y_i$  を用意 ( $X_i$  は  $m \times p$ )。 (2)  $Q = V^{-1}$  を決めて  $S = \sum X_i^\top Q X_i$ ,  $b = \sum X_i^\top Q y_i$  を計算。 (3)  $\hat{\theta} = S^{-1}b$ 。

■実装 R による実装例を以下に示す。

```
1 regression_multiple <- function(x, y, V = NULL, Q = NULL){
2   # y: n×1行列 (またはベクトル)
3   if (is.null(ncol(y))) {
4     y <- as.matrix(y, ncol = 1)
5   }
6   if (length(dim(x)) != 3) {
7     stop("x must be a 3-dimensional array")
8   }
9   n <- dim(x)[3]
10  m <- ncol(y)
11  p <- dim(x)[1] # 特徴量数
12  S <- matrix(0, p, p)
13  b <- matrix(0, p, 1)
14  T <- matrix(0, p, p)
15  if (is.null(V)) {
16    V <- diag(m)
17  }
```

```

18   if (is.null(Q)) {
19       Q <- solve(V)
20   }
21   # m=1 (スカラー出力) の場合はQ,Vをスカラー化
22   if (m == 1) {
23       Q <- as.numeric(Q)
24       V <- as.numeric(V)
25       for (i in 1:n) {
26           xi <- as.matrix(x[,i]) # (p,1)
27           S <- S + xi %*% t(xi) * Q
28           b <- b + xi * Q * y[i, 1]
29           T <- T + xi %*% t(xi) * V
30       }
31   } else {
32       for (i in 1:n) {
33           xi <- as.matrix(x[,i])
34           yi <- matrix(y[i, ], nrow = m, ncol = 1)
35           S <- S + t(xi) %*% Q %*% xi
36           b <- b + t(xi) %*% Q %*% yi
37           T <- T + t(xi) %*% V %*% xi
38       }
39   }
40   theta_hat <- solve(S) %*% b
41   err_cov_mat <- solve(S) %*% T %*% solve(S)
42   return(list(theta_hat = theta_hat, err_cov_mat = err_cov_mat, S = S, b = b, T = T
43   ))

```

### 3.2 課題5 (2次元出力：重み付き最小二乗法)

■課題の内容 入力  $x_i \in \mathbb{R}$  ( $i = 1, \dots, 1000$ ) に対し,  $m = 2$  次元出力の線形モデル

$$y_i = \phi(x_i)\theta + w_i, \quad \phi(x) = \begin{bmatrix} 1 & x \\ 1 & x^2 \end{bmatrix}, \quad w_i \sim \mathcal{N}(0, V), \quad V = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix},$$

を考える. ここで  $\theta \in \mathbb{R}^2$  を推定対象とする.

■推定法 最小二乗 (OLS) の推定値は

$$\hat{\theta}_N = \left( \sum_{i=1}^N \phi_i^\top \phi_i \right)^{-1} \sum_{i=1}^N \phi_i^\top y_i,$$

で与えられる ( $\phi_i = \phi(x_i)$ ). 観測雑音の共分散  $V$  が既知で, その影響を反映した重み付き最小二乗 (WLS) を用いると,  $V = W^{-2}$  を満たす  $W$  に対し

$$\hat{\theta}_N = \left( \sum_{i=1}^N \phi_i^\top W^{-2} \phi_i \right)^{-1} \sum_{i=1}^N \phi_i^\top W^{-2} y_i = \left( \sum_{i=1}^N \phi_i^\top Q \phi_i \right)^{-1} \sum_{i=1}^N \phi_i^\top Q y_i,$$

となる ( $Q = W^{-2} = V^{-1}$ ).

■実装 `datas/mmse_kadai5.csv` を読み込み,  $x$  から  $\phi(x) = \begin{bmatrix} 1 & x \\ 1 & x^2 \end{bmatrix}$  を構成して  $x \in \mathbb{R}^{2 \times 2 \times N}$  の3次元配列とし,  $y \in \mathbb{R}^{N \times 2}$  を観測行列とした. 推定は関数 `regression_multiple` を用い, **OLS**

は無重み ( $Q = I$ ), **WLS** は  $V = \text{diag}(100, 1)$  を渡し内部で  $Q = V^{-1}$  を用いる設定とした. また,  $N \in \{4, 8, 16, 32, 64, 128, 256, 512, 1000\}$  に対する  $\hat{\theta}_N$  の収束の様子を片対数 ( $x$  軸のみ対数) で可視化した.

■結果 全データ ( $N = 1000$ ) を用いた推定結果は次のとおりである.

**OLS**

$$\hat{\theta}_{\text{OLS}} = \begin{bmatrix} 2.994567 \\ -2.068971 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_{\text{OLS}}) = \begin{bmatrix} 5.7621 \times 10^{-4} & -1.5041 \times 10^{-4} \\ -1.5041 \times 10^{-4} & 2.9684 \times 10^{-4} \end{bmatrix}.$$

**WLS**

$$\hat{\theta}_{\text{WLS}} = \begin{bmatrix} 2.939084 \\ -1.986465 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_{\text{WLS}}) = \begin{bmatrix} 2.39993 \times 10^{-1} & -9.66541 \times 10^{-2} \\ -9.66541 \times 10^{-2} & 4.99126 \times 10^{-2} \end{bmatrix}.$$

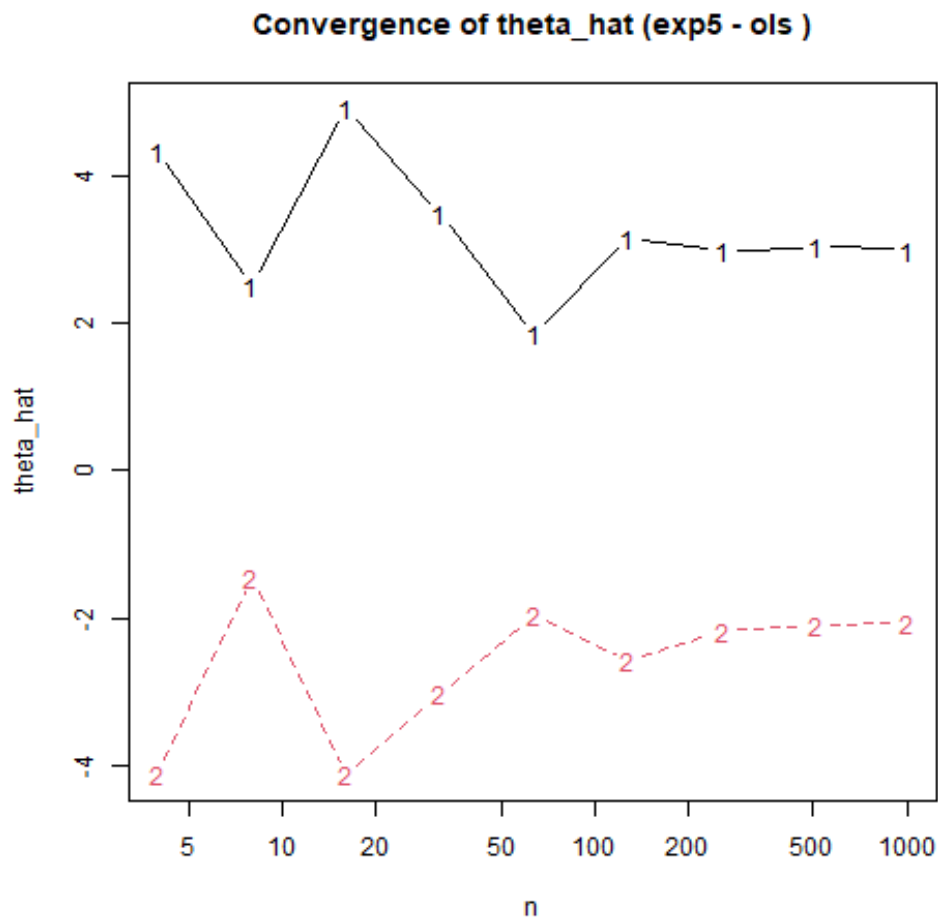


図 4:  $\hat{\theta}_N$  の収束 (OLS)

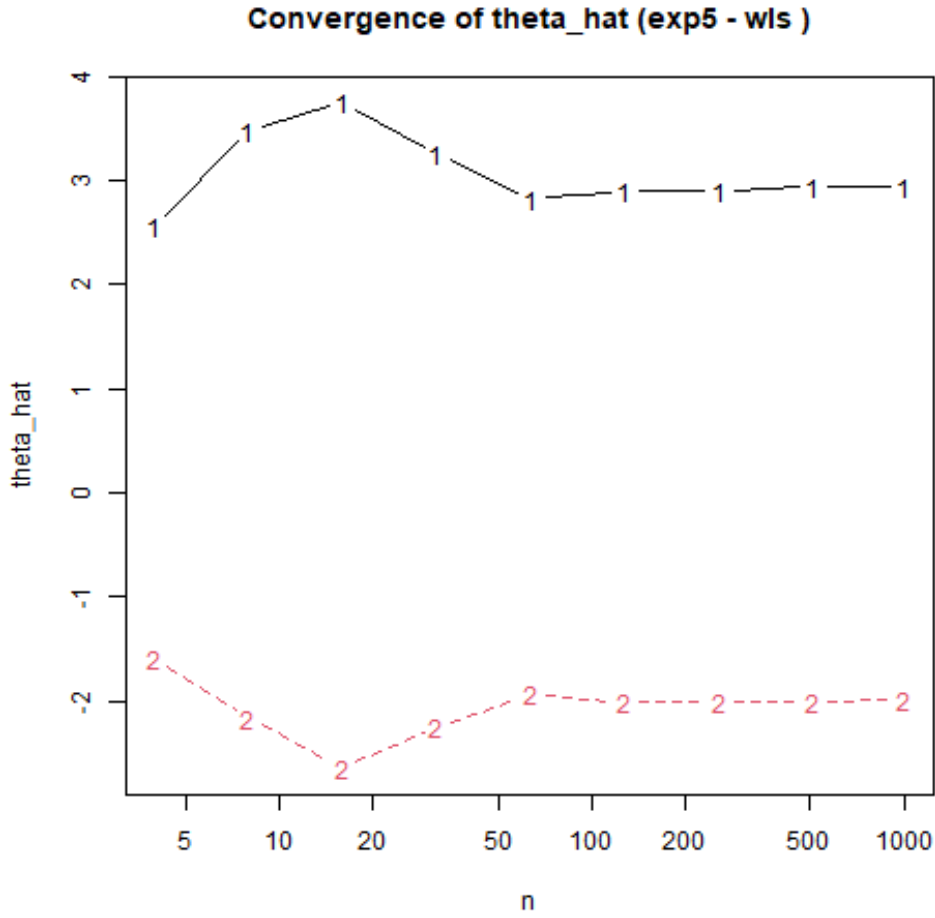


図 5:  $\hat{\theta}_N$  の収束 (WLS,  $V = \text{diag}(100, 1)$ )

## ■考察

### 3.3 課題 6 (2 次元出力・異分散 2 群に対する重み付き最小二乗)

■課題の内容 入力  $x_i \in \mathbb{R}$  に対し,

$$y_i = \phi(x_i)\theta + w_i, \quad \phi(x) = \begin{bmatrix} 1 & x \\ 1 & x^2 \end{bmatrix}, \quad \theta \in \mathbb{R}^2,$$

という  $m = 2$  次元出力の線形モデルを考える. 雑音は  $N$  サンプルのうち前半  $i = 1, \dots, \text{num}$  が  $w_i \sim \mathcal{N}(0, V_1)$ , 後半  $i = \text{num} + 1, \dots, N$  が  $w_i \sim \mathcal{N}(0, V_2)$  に従うとする. 本課題では  $V_1 = \text{diag}(100, 1)$ ,  $V_2 = \text{diag}(2, 1)$ ,  $\text{num} = 500$  を既知として推定を行う.

■推定法 (GLS/WLS)  $Q_k = V_k^{-1}$  ( $k = 1, 2$ ) とおくと, 一般化最小二乗 (WLS) の正規方程式は

$$S\hat{\theta}_N = b, \quad S = \sum_{i=1}^{\text{num}} \phi_i^\top Q_1 \phi_i + \sum_{i=\text{num}+1}^N \phi_i^\top Q_2 \phi_i, \quad b = \sum_{i=1}^{\text{num}} \phi_i^\top Q_1 y_i + \sum_{i=\text{num}+1}^N \phi_i^\top Q_2 y_i,$$

より

$$\hat{\theta}_N = S^{-1}b.$$

標本ベースの誤差共分散推定は（実装に合わせて）

$$\widehat{\text{Cov}}(\hat{\theta}_N) = S^{-1} T S^{-1}, \quad T = \sum_{i=1}^{\text{num}} \phi_i^\top V_1 \phi_i + \sum_{i=\text{num}+1}^N \phi_i^\top V_2 \phi_i.$$

なお OLS は  $Q_1 = Q_2 = I$  とした特別な場合である．

■実装 `datas/mmse_kadai6.csv` から  $x$  を読み、 $\phi(x)$  を用いて  $x \in \mathbb{R}^{2 \times 2 \times N}$ ,  $y \in \mathbb{R}^{N \times 2}$  を構成した．推定は関数 `regression_multiple_different_distributions` により、前半  $V_1$ 、後半  $V_2$  の重みで  $S, b, T$  を分割加算して  $\hat{\theta}_N$  と  $\widehat{\text{Cov}}(\hat{\theta}_N)$  を得た．また  $N$  を `ns = seq(4, 1000, length.out=20)` で走らせ、 $\hat{\theta}_N$  の収束の様子を折れ線で可視化した（ $x$  軸は線形目盛）．

■結果 全データ（ $N = 1000$ , `num` = 500）の推定結果は以下のとおりである．

OLS（比較）

$$\hat{\theta}_{\text{OLS}} = \begin{bmatrix} 3.188356 \\ -2.092183 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_{\text{OLS}}) = \begin{bmatrix} 5.692084 \times 10^{-4} & -1.402629 \times 10^{-4} \\ -1.402629 \times 10^{-4} & 2.842674 \times 10^{-4} \end{bmatrix}.$$

WLS（ $V_1 = \text{diag}(100, 1)$ ,  $V_2 = \text{diag}(2, 1)$ ）

$$\hat{\theta}_{\text{WLS}} = \begin{bmatrix} 2.994202 \\ -2.014699 \end{bmatrix}, \quad \widehat{\text{Cov}}(\hat{\theta}_{\text{WLS}}) = \begin{bmatrix} 6.019710 \times 10^{-2} & -2.227335 \times 10^{-2} \\ -2.227335 \times 10^{-2} & 1.276759 \times 10^{-2} \end{bmatrix}.$$

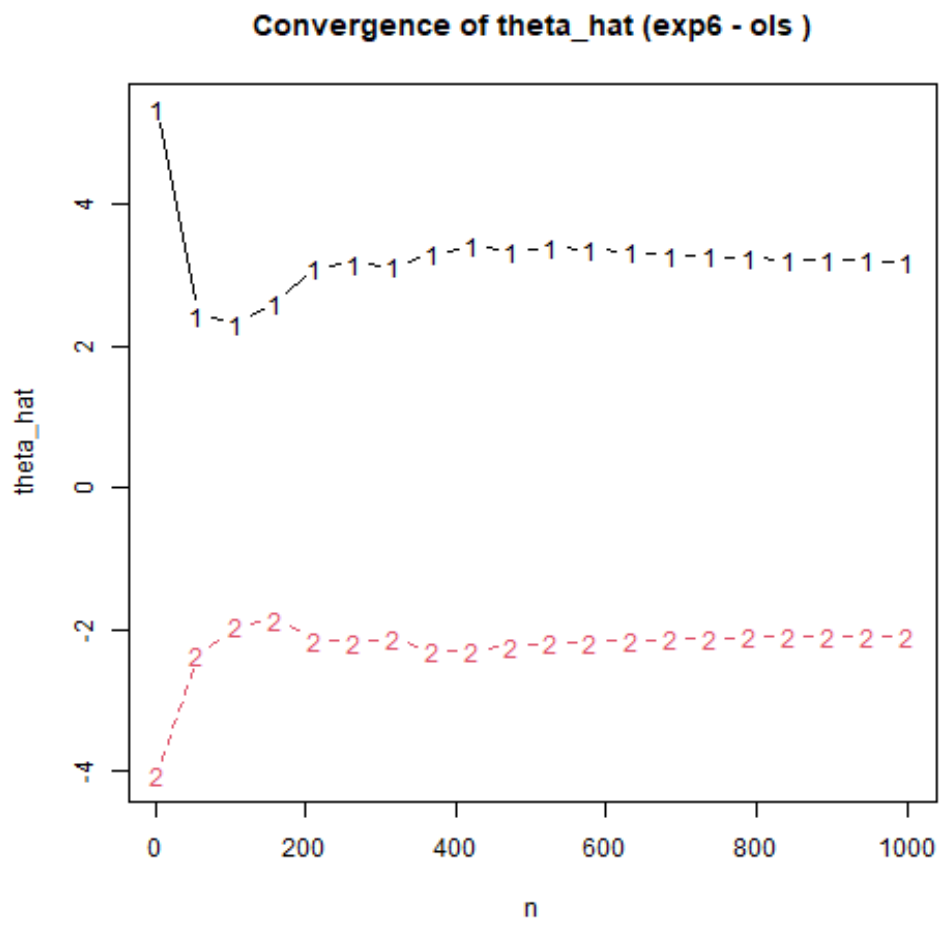


図 6:  $\hat{\theta}_N$  の収束 (OLS)

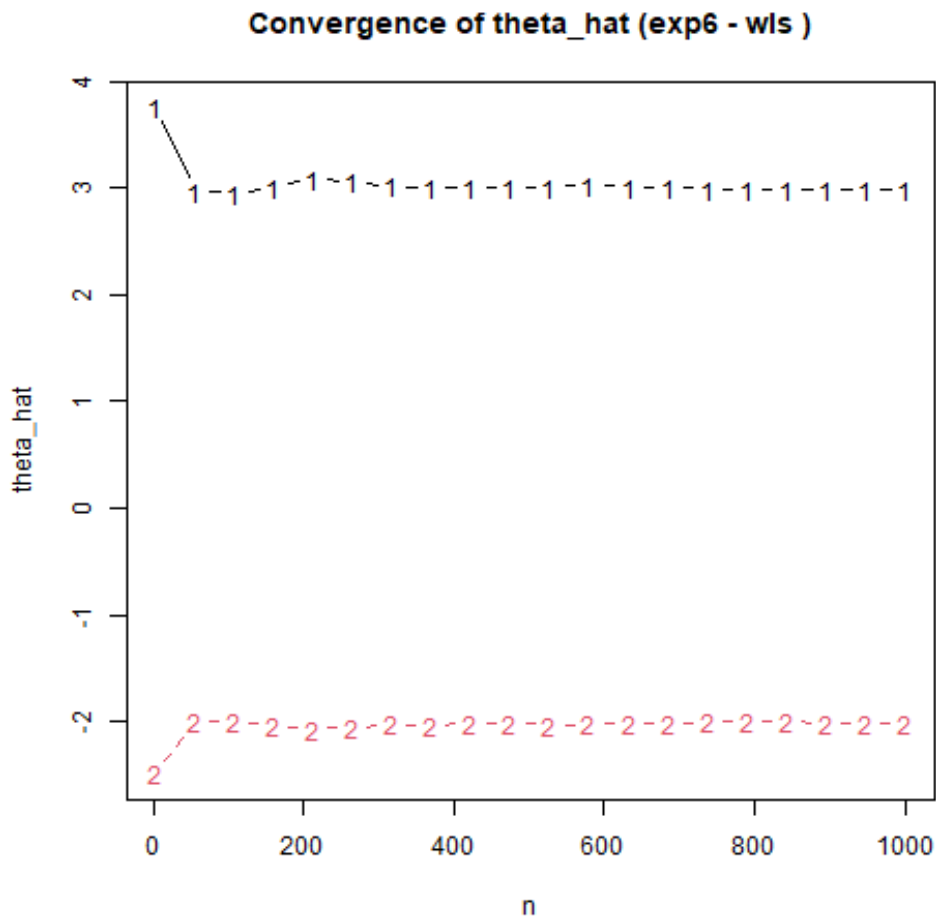


図 7:  $\hat{\theta}_N$  の収束 (WLS,  $V_1/V_2$  異分散)

## ■考察

# 4 推定値の合成と逐次最小二乗法

## 4.1 原理と方法

■推定値の合成 (Estimator Fusion)  $f(\theta, x) = \phi(x)\theta$  の線形回帰で,  $D_N$  と  $D'_M$  から得た推定  $\hat{\theta}_N = \Phi_N \sum_{i=1}^N \phi_i^\top y_i$ ,  $\hat{\theta}'_M = \Phi'_M \sum_{j=1}^M \phi'_j{}^\top y'_j$  ( $\Phi_N = (\sum \phi_i^\top \phi_i)^{-1}$ ) を持つとき, 結合推定は

$$\hat{\theta}_{N+M} = (\Phi_N^{-1} + \Phi'_M{}^{-1})^{-1} (\Phi_N^{-1} \hat{\theta}_N + \Phi'_M{}^{-1} \hat{\theta}'_M)$$

で与えられる。[1]

以下が R による実装例である。

```
1 # 既存バッチN=6000とM=4000の合成 (S =  $\Phi^{-1}$ ) を情報行列とする)
2 # S_6000, S_4000 はそれぞれ  $\Phi_6000^{-1}$ ,  $\Phi_4000^{-1}$ 
3 # theta_6000, theta_4000 は各バッチの推定ベクトル
4 theta_6000_4000 <- solve(S_6000 + S_4000) %*%
5 (S_6000 %*% theta_6000 + S_4000 %*% theta_4000)
```

■逐次最小二乗法 (RLS) 逆行列補題を用いると、バッチ解は次の再帰で更新できる：

$$\begin{aligned}\hat{\theta}_{N+1} &= \hat{\theta}_N + K_{N+1}(y_{N+1} - \phi_{N+1}\hat{\theta}_N), \\ K_{N+1} &= \Phi_N \phi_{N+1}^\top (I_m + \phi_{N+1} \Phi_N \phi_{N+1}^\top)^{-1}, \\ \Phi_{N+1} &= \Phi_N - K_{N+1} \phi_{N+1} \Phi_N.\end{aligned}$$

忘却係数  $\gamma \in (0, 1]$  を導入すると

$$\begin{aligned}K_{\gamma, N+1} &= \Phi_{\gamma, N} \phi_{N+1}^\top (\gamma I_m + \phi_{N+1} \Phi_{\gamma, N} \phi_{N+1}^\top)^{-1}, \\ \Phi_{\gamma, N+1} &= \gamma^{-1} (\Phi_{\gamma, N} - K_{\gamma, N+1} \phi_{N+1} \Phi_{\gamma, N}), \\ \hat{\theta}_{\gamma, N+1} &= \hat{\theta}_{\gamma, N} + K_{\gamma, N+1} (y_{N+1} - \phi_{N+1} \hat{\theta}_{\gamma, N}).\end{aligned}$$

[1]

以下が R による実装例である。

```
1 update_regression_mat <- function(Phi_N, phi_N_plus_1, theta_N, y_N_plus_1, gamma =
  1){
2   I_m <- diag(1) # m=1 想定。m>1 なら diag(nrow(phi_N_plus_1) %*% Phi_N %*% t(phi_N_plus_
  1))) に置換
3   # 中間計算
4   phi_T_Phi_phi <- phi_N_plus_1 %*% Phi_N %*% t(phi_N_plus_1)
5   inv_term <- solve( (gamma * I_m) + phi_T_Phi_phi )
6   K <- gamma * (Phi_N %*% t(phi_N_plus_1) %*% inv_term)
7   # 更新
8   Phi_N_plus_1 <- (Phi_N - K %*% phi_N_plus_1 %*% Phi_N) / gamma
9   theta_N_plus_1 <- theta_N + K %*% (y_N_plus_1 - phi_N_plus_1 %*% theta_N)
10  list(Phi_N_plus_1 = Phi_N_plus_1, theta_N_plus_1 = theta_N_plus_1)
11 }
```

$\Phi_0 = \varepsilon^{-1} I$  ( $\varepsilon$  は小さい正の実数) で初期化し、逐次更新する。各ステップの計算は  $\dim(\theta) = p$  に対して  $O(p^2)$  程度で済む。[1]

## 4.2 課題 7 (推定値の合成の検証：2 分割データの統合)

■課題の内容  $N = 10000$  のデータを前半 6000 と後半 4000 に分割し、各ブロックで OLS 推定を行ったのち、情報行列の加法性

$$S_N = \sum_{i=1}^N \varphi_i \varphi_i^\top$$

を用いた合成推定

$$\hat{\theta}_{\text{fuse}} = (S_{6000} + S_{4000})^{-1} (S_{6000} \hat{\theta}_{6000} + S_{4000} \hat{\theta}_{4000})$$

が、全データ一括推定  $\hat{\theta}_{10000}$  と一致することを確認する。ここでモデルは

$$y_i = \varphi(x_i)^\top \theta + w_i, \quad \varphi(x) = \begin{bmatrix} 1 \\ \exp(-\frac{(x-1)^2}{2}) \\ \exp(-(x+1)^2) \end{bmatrix}, \quad \theta \in \mathbb{R}^3.$$

理論上、独立同分布で  $V = \sigma^2$  のとき  $\hat{\theta}_{\text{fuse}} = \hat{\theta}_{10000}$  となる。[1]



■実装 分割データから  $S_{6000}, \hat{\theta}_{6000}, S_{4000}, \hat{\theta}_{4000}$  を得て、合成推定値を計算し、全データ一括推定と比較する。

```

1 # 事前に x, y, x_6000, y_6000, x_4000, y_4000 を用意済み
2
3 exp7 <- function(){
4   res6000 <- regression_multiple(x_6000, y_6000)
5   res4000 <- regression_multiple(x_4000, y_4000)
6   res10000 <- regression_multiple(x, y)
7
8   S_6000 <- res6000$S
9   S_4000 <- res4000$S
10  theta_6000 <- res6000$theta_hat
11  theta_4000 <- res4000$theta_hat
12
13  # 推定値の合成 (情報行列の和)
14  theta_6000_4000 <- solve(S_6000 + S_4000) %*%
15    (S_6000 %*% theta_6000 + S_4000 %*% theta_4000)
16
17  cat("S6000:\n"); print(S_6000)
18  cat("S4000:\n"); print(S_4000)
19  cat("theta6000:\n"); print(theta_6000)
20  cat("theta4000:\n"); print(theta_4000)
21  cat("theta_fuse:\n"); print(theta_6000_4000)
22
23  theta_10000 <- res10000$theta_hat
24  cat("theta10000:\n"); print(theta_10000)
25
26  # 一致性の確認 (数値誤差内)
27  cat("all.equal(theta_fuse, theta10000): ",
28      all.equal(drop(theta_6000_4000), drop(theta_10000),
29                tolerance = 1e-10), "\n")
30 }
31
32 exp7()

```

■結果 一例として、実行時に次を得た。

$$S_{6000} = \begin{bmatrix} 6000.000 & 1505.370 & 1033.499 \\ 1505.370 & 1063.211 & 226.401 \\ 1033.499 & 226.401 & 720.562 \end{bmatrix}, \quad S_{4000} = \begin{bmatrix} 4000.000 & 971.151 & 716.318 \\ 971.151 & 679.477 & 149.749 \\ 716.318 & 149.749 & 510.224 \end{bmatrix},$$

$$\hat{\theta}_{6000} = \begin{bmatrix} -0.003879385 \\ 3.010714895 \\ -1.989434350 \end{bmatrix}, \quad \hat{\theta}_{4000} = \begin{bmatrix} -0.02537589 \\ 3.03489937 \\ -1.97772731 \end{bmatrix}, \quad \hat{\theta}_{\text{fuse}} = \begin{bmatrix} -0.0124955 \\ 3.0204300 \\ -1.9848692 \end{bmatrix}.$$

全データ一括推定  $\hat{\theta}_{10000}$  と  $\hat{\theta}_{\text{fuse}}$  は数値誤差内で一致した (コードで `all.equal` により検証)。

■考察

### 4.3 課題 7 (推定値の合成の検証：非線形基底・分割データ)

#### ■課題の内容 基底

$$\varphi(x) = \begin{bmatrix} 1 \\ \exp(-\frac{(x-1)^2}{2}) \\ \exp(-(x+1)^2) \end{bmatrix}, \quad y_i = \varphi(x_i)^\top \theta + w_i, \quad \theta \in \mathbb{R}^3$$

で  $N = 10000$  個のデータを前半 6000 と後半 4000 に分割する。各ブロックの推定値  $(\hat{\theta}_{6000}, \hat{\theta}_{4000})$  と情報行列  $S_k = \sum_{i \in \text{block } k} \varphi_i \varphi_i^\top$  を用いて

$$\hat{\theta}_{\text{fuse}} = (S_{6000} + S_{4000})^{-1} (S_{6000} \hat{\theta}_{6000} + S_{4000} \hat{\theta}_{4000})$$

を計算し、全データ一括推定  $\hat{\theta}_{10000}$  と照合する。[1]

#### ■実装 (OLS 版)

```
1 data <- read.csv("datas/mmse_kadai7.csv", header=FALSE, col.names=c("x","y"))
2 x0 <- rep(1, nrow(data))
3 x1 <- exp(-(data$x - 1)^2 / 2)
4 x2 <- exp(-(data$x + 1)^2)
5 n <- nrow(data)
6 x <- array(0, dim=c(3,1,n))
7 for(i in 1:n){ x[, ,i] <- matrix(c(x0[i], x1[i], x2[i]), 3, 1) }
8 y <- as.matrix(data[, "y"])
9
10 x_6000 <- x[, ,1:6000, drop=FALSE]; y_6000 <- y[1:6000, , drop=FALSE]
11 x_4000 <- x[, ,6001:10000, drop=FALSE]; y_4000 <- y[6001:10000, , drop=FALSE]
12
13 exp7 <- function(){
14   r6 <- regression_multiple(x_6000, y_6000) # returns $$S$ and $theta_hat$
15   r4 <- regression_multiple(x_4000, y_4000)
16   rall <- regression_multiple(x, y)
17
18   S_6000 <- r6$S; S_4000 <- r4$S
19   th6 <- r6$theta_hat; th4 <- r4$theta_hat; th_all <- rall$theta_hat
20
21   th_fuse <- solve(S_6000 + S_4000) %*% (S_6000 %*% th6 + S_4000 %*% th4)
22
23   print(th6); print(th4); print(th_fuse); print(th_all)
24   # 一致検証
25   print(all.equal(drop(th_fuse), drop(th_all), tolerance=1e-12))
26 }
27 exp7()
```

#### ■実装 (推定分散による重み付き合成) 各ブロックの誤差分散を

$$\hat{\sigma}_k^2 = \frac{\text{RSS}_k}{N_k - p}, \quad p = 3$$

で推定し、 $Q_k = \hat{\sigma}_k^{-2}$  を用いた WLS 情報行列  $S_k = \sum_{i \in k} \varphi_i Q_k \varphi_i^\top = Q_k \sum_{i \in k} \varphi_i \varphi_i^\top$  で合成する ( $k \in \{6000, 4000\}$ )。コードは次のとおり。

```
1 x_mat <- t(matrix(x, nrow=3, ncol=n))
2 x_6000_mat <- t(matrix(x_6000, nrow=3, ncol=6000))
```

```

3 x_4000_mat <- t(matrix(x_4000, nrow=3, ncol=4000))
4
5 exp8 <- function(){
6   # OLS 推定
7   th_all <- regression_multiple(x, y)$theta_hat
8   th_6000 <- regression_multiple(x_6000, y_6000)$theta_hat
9   th_4000 <- regression_multiple(x_4000, y_4000)$theta_hat
10
11  # 分散推定 (MSE)
12  V_hat_full <- sum((y - x_mat %*% th_all )^2) /(nrow(x_mat) - ncol(
13    x_mat))
14  V_hat_6000 <- sum((y_6000 - x_6000_mat %*% th_6000)^2) /(nrow(x_6000_mat) - ncol(
15    x_6000_mat))
16  V_hat_4000 <- sum((y_4000 - x_4000_mat %*% th_4000)^2) /(nrow(x_4000_mat) - ncol(
17    x_4000_mat))
18
19  # 各ブロックの WLS 情報行列と推定
20  r6 <- regression_multiple(x_6000, y_6000, V=V_hat_6000)
21  r4 <- regression_multiple(x_4000, y_4000, V=V_hat_4000)
22  S_6000 <- r6$S; S_4000 <- r4$S
23  th6 <- r6$theta_hat; th4 <- r4$theta_hat
24
25  # 合成推定 (WLS 版)
26  th_fuse <- solve(S_6000 + S_4000) %*% (S_6000 %*% th6 + S_4000 %*% th4)
27
28  # 参考：全データにスカラー重み V_hat_full をかけた WLS (OLS と同値)
29  th_full_wls <- regression_multiple(x, y, V=V_hat_full)$theta_hat
30
31  print(th6); print(th4); print(th_fuse); print(th_full_wls)
32 }
33 exp8()

```

■結果 実行例（あなたの出力）：

$$\hat{\theta}_{6000} = \begin{bmatrix} -0.003879385 \\ 3.010714895 \\ -1.989434350 \end{bmatrix}, \quad \hat{\theta}_{4000} = \begin{bmatrix} -0.02537589 \\ 3.03489937 \\ -1.97772731 \end{bmatrix}.$$

WLS 合成推定と全データ一括 WLS（スカラー重み）：

$$\hat{\theta}_{\text{fuse}} = \begin{bmatrix} -0.01245376 \\ 3.02038299 \\ -1.98489169 \end{bmatrix}, \quad \hat{\theta}_{10000} = \begin{bmatrix} -0.0124955 \\ 3.0204300 \\ -1.9848692 \end{bmatrix}.$$

差分は  $\Delta\theta \approx (4.17 \times 10^{-5}, -4.70 \times 10^{-5}, -2.25 \times 10^{-5})^\top$ 。ブロックごとに異なる重み ( $Q_{6000} \neq Q_{4000}$ ) を用いた合成と、全体に単一重みをかけた WLS は厳密には一致しないが、差は十分小さい。  
[1]

■考察

#### 4.4 課題 9（逐次最小二乗法によるシステム同定）

■モデル 離散化されたバネ・マス・ダンパ系 ( $M = 2, D = 1, K = 3, \Delta t = 0.01$ )

$$y_k = a_1 y_{k-1} + a_2 y_{k-2} + b F_{k-2} + w_k, \quad \phi_k = [y_{k-1} \quad y_{k-2} \quad F_{k-2}]^T,$$

$$a_1 = 2 - \frac{D}{M}\Delta t = 1.995, \quad a_2 = -(1 - \frac{D}{M}\Delta t + \frac{K}{M}\Delta t^2) = -0.99515, \quad b = \frac{\Delta t^2}{M} = 5.0 \times 10^{-5}.$$

$w_k \sim \text{Uniform}[-1, 1]$ 。RLS 更新は §3.3 の式に従う（忘却なし  $\gamma = 1$ ） [1]。

#### ■使用コード（共通）

```

1 w <- function(){ runif(1, min = -1, max = 1) }
2
3 update_tick <- function(y_k_1, y_k_2, F_k_2,
4                         M=2, D=1, K=3, delta_t=0.01){
5   (2 - (D/M)*delta_t) * y_k_1 -
6   (1 - (D/M)*delta_t + (K/M)*delta_t^2) * y_k_2 +
7   (delta_t^2/M) * F_k_2 + w()
8 }
9
10 # RLS (§3.3 の式に一致)
11 update_regression_mat <- function(Phi_N, phi_N_plus_1, theta_N, y_N_plus_1, gamma=1){
12   I_m <- diag(1)
13   phi_T_Phi_phi <- phi_N_plus_1 %*% Phi_N %*% t(phi_N_plus_1)
14   inv_term <- solve((gamma * I_m) + phi_T_Phi_phi)
15   K <- gamma * (Phi_N %*% t(phi_N_plus_1) %*% inv_term)
16   Phi_N_plus_1 <- (Phi_N - K %*% phi_N_plus_1 %*% Phi_N) / gamma
17   theta_N_plus_1 <- theta_N + K %*% (y_N_plus_1 - phi_N_plus_1 %*% theta_N)
18   list(Phi_N_plus_1=Phi_N_plus_1, theta_N_plus_1=theta_N_plus_1)
19 }

```

#### (1) 入力 $F_k \equiv 1$ （定数入力）

■内容  $F_k = 1$  を一定とし、 $k = 1, \dots, 10000$  で RLS により  $\theta = [a_1, a_2, b]^\top$  を推定する。 [1]

■結果 実行出力：

$$\hat{\theta} = \begin{bmatrix} 1.99515410 \\ -0.99532391 \\ 2.15087 \times 10^{-3} \end{bmatrix}.$$

■考察  $a_1, a_2$  は真値に近い。 $b$  は真値  $5 \times 10^{-5}$  に対して過大。定数入力で効果が極小かつ雑音が大きく、入力励起が不十分。 $F$  の情報が弱く  $b$  の識別性が低い。

#### ■コード

```

1 exp9_1 <- function(){
2   y_k <- y_k_1 <- y_k_2 <- 0; F_k_2 <- 1
3   Phi <- diag(3) * 1000; theta <- matrix(0, nrow=3)
4   for(i in 1:10000){
5     y_k_2 <- y_k_1; y_k_1 <- y_k
6     y_k <- update_tick(y_k_1, y_k_2, F_k_2)
7     xi <- matrix(c(y_k_1, y_k_2, F_k_2), nrow=1)
8     upd <- update_regression_mat(Phi, xi, theta, y_k)
9     Phi <- upd$Phi_N_plus_1; theta <- upd$theta_N_plus_1
10  }
11  print(theta)
12 }

```

#### (2) 入力 $F_k = \sin(\pi k/5)$ （小振幅正弦）

■内容  $F$  を低振幅の正弦波にして同様に推定。 [1]

■結果 実行出力：

$$\hat{\theta} = \begin{bmatrix} 1.995721237 \\ -0.995895110 \\ 2.637203 \times 10^{-3} \end{bmatrix}.$$

■考察  $F$  は時間変化するが振幅が小さく、 $b$  の寄与は依然として雑音に埋もれる。 $b$  は真値より二桁以上大きい。入力励起の大きさが推定精度の律速。

■コード

```
1 exp9_2 <- function(){
2   y_k <- y_k_1 <- y_k_2 <- 0
3   Phi <- diag(3) * 1000; theta <- matrix(0, nrow=3)
4   for(i in 1:10000){
5     F_k_2 <- sin(pi * i / 5)
6     y_k_2 <- y_k_1; y_k_1 <- y_k
7     y_k <- update_tick(y_k_1, y_k_2, F_k_2)
8     xi <- matrix(c(y_k_1, y_k_2, F_k_2), nrow=1)
9     upd <- update_regression_mat(Phi, xi, theta, y_k)
10    Phi <- upd$Phi_N_plus_1; theta <- upd$theta_N_plus_1
11  }
12  print(theta)
13 }
```

(3) 入力  $F_k = 10000 \sin(\pi k/5)$  (大振幅正弦)

■内容 入力振幅を  $10^4$  倍して強い励起を与える。[1]

■結果 実行出力：

$$\hat{\theta} = \begin{bmatrix} 1.994996 \\ -0.995125 \\ 4.883587 \times 10^{-5} \end{bmatrix}.$$

■考察  $b$  が真値  $5.0 \times 10^{-5}$  に一致。十分な励起で入出力比が改善し、 $F$  の係数が正しく同定できる。 $a_1, a_2$  も誤差が小さい。RLS はオンラインに有効だが、入力の持続励起とスケール設計が必要。

■コード

```
1 exp9_3 <- function(){
2   y_k <- y_k_1 <- y_k_2 <- 0
3   Phi <- diag(3) * 1000; theta <- matrix(0, nrow=3)
4   for(i in 1:10000){
5     F_k_2 <- 10000 * sin(pi * i / 5)
6     y_k_2 <- y_k_1; y_k_1 <- y_k
7     y_k <- update_tick(y_k_1, y_k_2, F_k_2)
8     xi <- matrix(c(y_k_1, y_k_2, F_k_2), nrow=1)
9     upd <- update_regression_mat(Phi, xi, theta, y_k)
10    Phi <- upd$Phi_N_plus_1; theta <- upd$theta_N_plus_1
11  }
12  print(theta)
13 }
```

## 4.5 課題 10 (非定常時系列の追従：忘却係数付き RLS)

■内容 非定常信号

$$y_k = \theta_k + w_k, \quad \theta_k = \sin(10^{-4}k), \quad w_k \in \{-1, +1\}$$

を 1 次元回帰  $\phi_k \equiv [1]$  で逐次推定する。忘却係数  $\gamma = 0.99$  を用いる。[1]

■方法 忘却付き RLS の更新 ( $m = 1$ ) :

$$\begin{aligned} K_k &= \Phi_{k-1} (\gamma + \phi_k \Phi_{k-1} \phi_k^\top)^{-1} \phi_k^\top, \\ \theta_k^{\text{hat}} &= \theta_{k-1}^{\text{hat}} + K_k (y_k - \phi_k \theta_{k-1}^{\text{hat}}), \\ \Phi_k &= \gamma^{-1} (\Phi_{k-1} - K_k \phi_k \Phi_{k-1}), \end{aligned}$$

初期化は  $\theta_0^{\text{hat}} = 0$ ,  $\Phi_0 = 1000$ 。  $\gamma = 0.99$  は有効窓長の目安  $1/(1 - \gamma) \approx 100$  に相当。[1]

■実装

```

1 exp10 <- function(){
2   y_k <- 0
3   Phi <- matrix(1000, 1, 1)
4   theta_hat <- matrix(0, 1, 1)
5   theta_hats <- c()
6   for(k in 1:10000){
7     w_k <- sample(c(-1, 1), 1)           # ノイズ
8     y_k <- sin(0.0001 * k) + w_k        # 観測
9     xi <- matrix(1, 1, 1)               #  $\phi_k = 1$ 
10    upd <- update_regression_mat(Phi, xi, theta_hat, y_k, gamma=0.99)
11    Phi <- upd$Phi_N_plus_1
12    theta_hat <- upd$theta_N_plus_1
13    theta_hats <- rbind(theta_hats, theta_hat)
14  }
15  dir.create("graphs", showWarnings=FALSE)
16  png("graphs/task10.png")
17  plot(1:10000, theta_hats, type="l", ylim=c(-1.5, 1.5),
18       xlab="k", ylab="theta_hat",
19       main="Estimation of non-stationary time series (exp10)")
20  lines(1:10000, sin(0.0001 * (1:10000)))
21  legend("topright", legend=c("Estimated theta_hat", "True theta"), lty=1)
22  dev.off()
23 }
24 exp10()
```

■結果 図 8。推定  $\theta_k^{\text{hat}}$  は真値  $\sin(10^{-4}k)$  を平滑追従。ノイズ  $\pm 1$  のため短周期の揺れが残る。

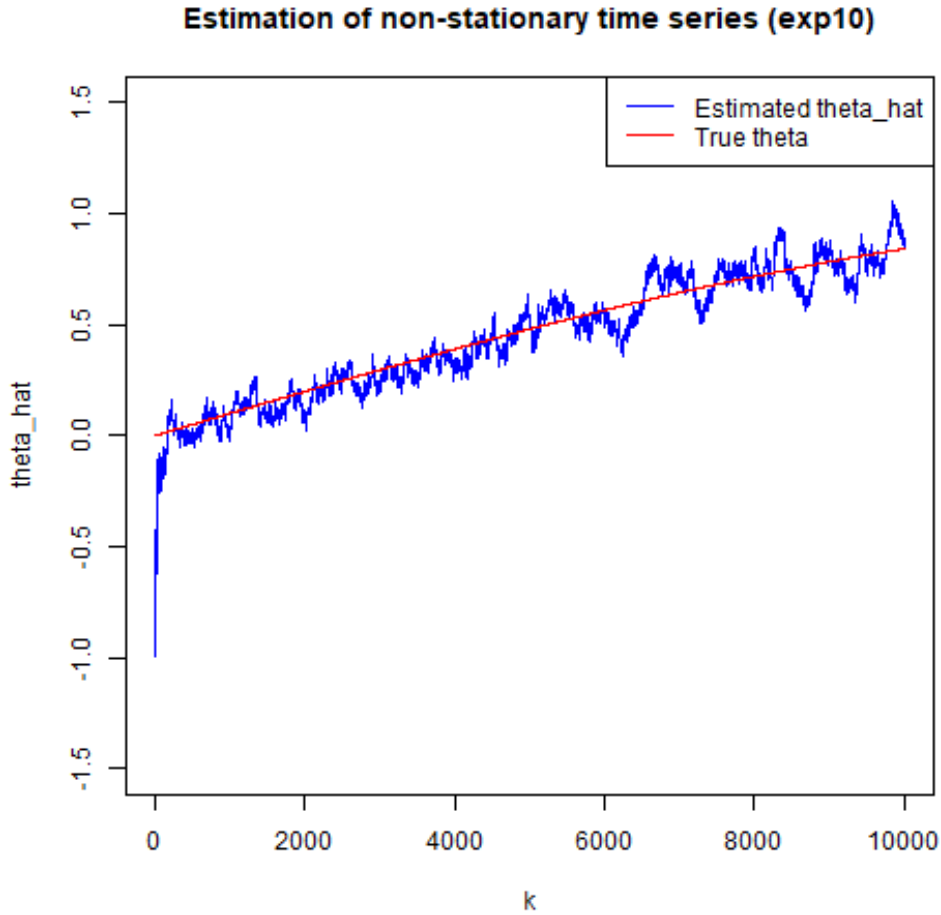


図 8: 忘却係数付き RLS による非定常平均の追従 ( $\gamma = 0.99$ )

## ■考察

# 5 カルマンフィルタ、カルマン smoother

## 5.1 原理と方法

■モデル 一次元の線形状態空間モデルを用いる：

$$\theta_k = a_k \theta_{k-1} + v_k, \quad (3.40)$$

$$y_k = c_k \theta_k + w_k, \quad (3.41)$$

ここで  $v_k \sim \mathcal{N}(0, \sigma_v^2)$ ,  $w_k \sim \mathcal{N}(0, \sigma_w^2)$  は互いに独立で白色,  $a_k, c_k$  は既知とする [1]。目的は「時刻  $k$  までの観測  $\{y_\ell\}_{\ell=1}^k$  に基づく  $\theta_k$  の最小平均二乗誤差 (MMSE) 推定」を行うこと [1]。

以下はこのモデルに基づくデータ生成の R による実装例である。

### Listing 1: データ生成関数

```
1 # データ生成関数 (exp11 と exp12 で共通使用)
2 generate_kalman_data <- function(seed = 123, N = 100, a_k = 0.9, c_k = 2,
```

```

3                                     sigma2_v = 1, sigma2_w = 1) {
4   # 乱数シードを設定
5   set.seed(seed)
6
7   # 初期値
8   theta_0 <- rnorm(1, mean = 3, sd = sqrt(2)) #  $\theta_0 \sim N(3, 2)$ 
9   V_0 <- 2
10
11  # データ格納用
12  theta_true <- numeric(N)
13  y_obs <- numeric(N)
14
15  # 初期化
16  theta_k_1 <- theta_0
17
18  # データ生成
19  for (k in 1:N) {
20    # 真の状態遷移 (3.40):  $\theta_k = a_k * \theta_{k-1} + v_k$ 
21    v_k <- rnorm(1, mean = 0, sd = sqrt(sigma2_v))
22    theta_k <- a_k * theta_k_1 + v_k
23    theta_true[k] <- theta_k
24
25    # 観測 (3.41):  $y_k = c_k * \theta_k + w_k$ 
26    w_k <- rnorm(1, mean = 0, sd = sqrt(sigma2_w))
27    y_k <- c_k * theta_k + w_k
28    y_obs[k] <- y_k
29
30    # 次のステップへ
31    theta_k_1 <- theta_k
32  }
33
34  # データを返す
35  return(list(
36    theta_true = theta_true,
37    y_obs = y_obs,
38    theta_0 = theta_0,
39    V_0 = V_0,
40    a_k = a_k,
41    c_k = c_k,
42    sigma2_v = sigma2_v,
43    sigma2_w = sigma2_w,
44    N = N
45  ))
46 }

```

■Kalman フィルタ 平方完成による最適ゲイン  $F_k$  と分散更新を導くと、予測分散  $X_k$ ，事後分散  $V_k$ ，推定値  $\hat{\theta}_k$  は

$$\hat{\theta}_k = a_k \hat{\theta}_{k-1} + F_k (y_k - c_k a_k \hat{\theta}_{k-1}), \quad (3.36)$$

$$X_k = a_k^2 V_{k-1} + \sigma_v^2, \quad (3.37)$$

$$V_k = \frac{\sigma_w^2 X_k}{c_k^2 X_k + \sigma_w^2}, \quad (3.38)$$

$$F_k = \frac{c_k X_k}{c_k^2 X_k + \sigma_w^2}. \quad (3.39)$$



これが Kalman フィルタであり、推定誤差分散を最小にする [1]。初期条件は  $\hat{\theta}_0 = \bar{\theta}$ ,  $V_0 = P$  とする [1]。

以下は、Kalman フィルタの R による実装例である。

**Listing 2:** Kalman フィルタ更新 (式 (3.36) – (3.39))

```
1 update_kalman_filter <- function(theta_k_1, V_k_1, y_k, a_k, c_k, sigma2_v, sigma2_w)
2 {
3   # (3.37) 予測分散
4   X_k <- a_k^2 * V_k_1 + sigma2_v
5   # (3.39) カルマンゲイン
6   F_k <- (c_k * X_k) / (c_k^2 * X_k + sigma2_w)
7   # (3.36) 推定値更新
8   theta_k <- a_k * theta_k_1 + F_k * (y_k - c_k * a_k * theta_k_1)
9   # (3.38) 事後分散更新
10  V_k <- (sigma2_w * X_k) / (c_k^2 * X_k + sigma2_w)
11  list(theta_k = theta_k, V_k = V_k, X_k = X_k, F_k = F_k)
12 }
```

■Kalman スムーザ 全データ  $\{y_1, \dots, y_N\}$  を用いて各  $\theta_k$  を後処理で精密化する。Rauch – Tung – Striebel (RTS) 型の退行更新は

$$\hat{\theta}_k^s = \hat{\theta}_k + g_k(\hat{\theta}_{k+1}^s - a_k \hat{\theta}_k), \quad (3.42)$$

$$g_k = \frac{a_k V_k}{X_{k+1}}, \quad (3.43)$$

$$V_k^s = V_k + g_k^2(V_{k+1}^s - X_{k+1}), \quad (3.44)$$

終端条件は  $\hat{\theta}_N^s = \hat{\theta}_N$ ,  $V_N^s = V_N$ 。  $\{\hat{\theta}_k^s\}$  は時刻  $N$  までの全情報に基づく MMSE 推定となる [1]。

以下は、Kalman スムーザの R による実装例である。

**Listing 3:** Kalman スムーザ更新 (式 (3.42) – (3.44))

```
1 update_kalman_smoother <- function(theta_k, theta_k_plus_1_s, V_k, V_k_plus_1_s, X_k_
2   plus_1, a_k) {
3   # (3.43) 後向きゲイン
4   g_k <- a_k * V_k / X_k_plus_1
5   # (3.42) 推定値の後向き補正
6   theta_k_s <- theta_k + g_k * (theta_k_plus_1_s - a_k * theta_k)
7   # (3.44) 分散の後向き補正
8   V_k_s <- V_k + g_k^2 * (V_k_plus_1_s - X_k_plus_1)
9   list(theta_k_s = theta_k_s, V_k_s = V_k_s, g_k = g_k)
10 }
```

## 5.2 課題 11 (カルマンフィルタ：フォワード)

■内容 一次元状態空間モデルを対象とする。

$$\theta_k = a_k \theta_{k-1} + v_{k-1}, \quad v_{k-1} \sim \mathcal{N}(0, \sigma_v^2), \quad (14)$$

$$y_k = c_k \theta_k + w_k, \quad w_k \sim \mathcal{N}(0, \sigma_w^2). \quad (15)$$

初期推定値は  $\hat{\theta}_0 = \theta_0$ , 分散  $V_0$  を与える. 評価は平均二乗誤差

$$\text{MSE} = \frac{1}{N} \sum_{k=1}^N (\theta_k^{\text{true}} - \hat{\theta}_k)^2 \quad (16)$$

で行う.

■方法 時刻  $k$  ごとに予測と更新を行う.

$$X_k = a_k^2 V_{k-1} + \sigma_v^2, \quad (17)$$

$$F_k = \frac{c_k X_k}{c_k^2 X_k + \sigma_w^2}, \quad (18)$$

$$\hat{\theta}_k = a_k \hat{\theta}_{k-1} + F_k (y_k - c_k a_k \hat{\theta}_{k-1}), \quad (19)$$

$$V_k = (1 - F_k c_k) X_k. \quad (20)$$

出力は  $\{\hat{\theta}_k\}_{k=1}^N$  と  $\{V_k\}_{k=1}^N$ .

■実装 関数 `exp11(data)` を用いる. 各  $k$  で `update_kalman_filter` を呼び,  $X_k, F_k, \hat{\theta}_k, V_k$  を更新する. 推定系列と真値を重ね描画し `graphs/task11.png` に保存する.

■結果 データ生成と実行条件は

```
kalman_data <- generate_kalman_data(seed = 42, N = 100,
                                     a_k = 0.9, c_k = 2,
                                     sigma2_v = 1, sigma2_w = 1)
result11 <- exp11(kalman_data)
```

である. MSE は 0.1935684 となった. 図 9 に真値と推定の系列を示す.

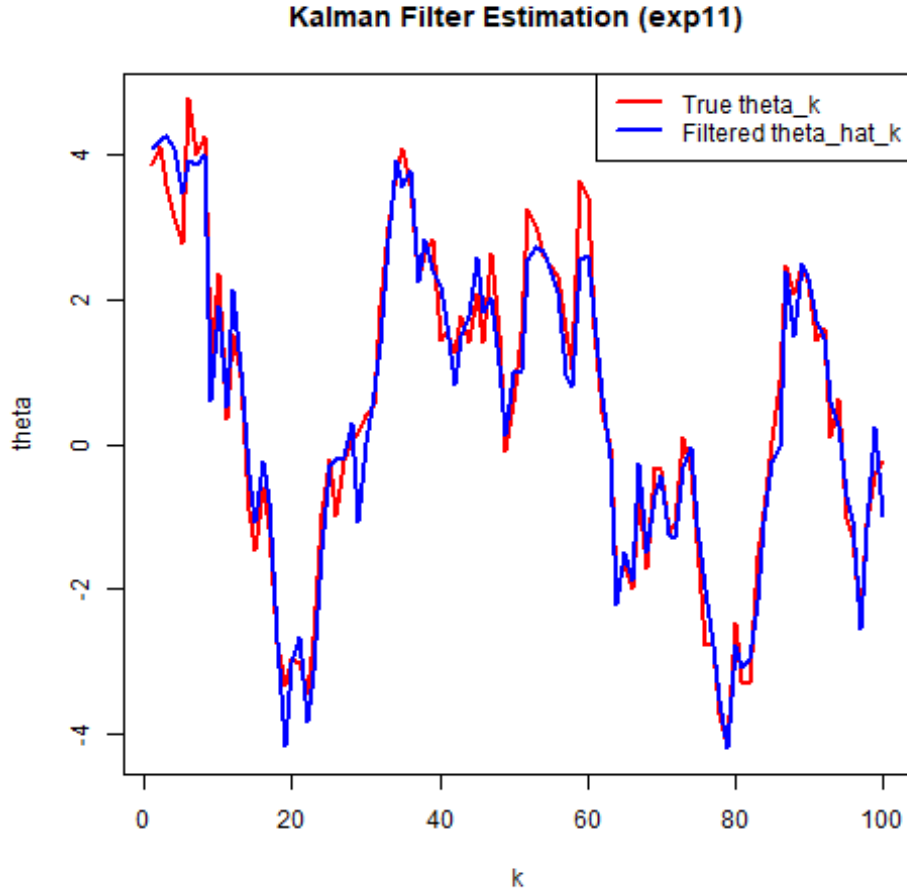


図 9: カルマンフィルタ（フォワード）による推定結果（赤：真値，青：推定）

## ■考察

### 5.3 課題 12（カルマン smoother）

#### ■内容 一次元状態空間モデル

$$\theta_k = a_k \theta_{k-1} + v_{k-1}, \quad v_{k-1} \sim \mathcal{N}(0, \sigma_v^2), \quad (21)$$

$$y_k = c_k \theta_k + w_k, \quad w_k \sim \mathcal{N}(0, \sigma_w^2) \quad (22)$$

に対し，時刻  $1:N$  の全観測を用いて各時刻の状態を後向きに精緻化する．

#### ■方法 まず課題 11 と同じ前向きフィルタで

$$X_k = a_k^2 V_{k-1} + \sigma_v^2, \quad F_k = \frac{c_k X_k}{c_k^2 X_k + \sigma_w^2}, \quad (23)$$

$$\hat{\theta}_k = a_k \hat{\theta}_{k-1} + F_k (y_k - c_k a_k \hat{\theta}_{k-1}), \quad V_k = (1 - F_k c_k) X_k \quad (24)$$

を得る．後向きでは RTS smoother を用いる（スカラー）．平滑化ゲイン

$$J_k = \frac{V_k a_k}{X_{k+1}}, \quad (25)$$

更新式

$$\hat{\theta}_k^s = \hat{\theta}_k + J_k(\hat{\theta}_{k+1}^s - a_k \hat{\theta}_k), \quad (26)$$

$$V_k^s = V_k + J_k^2(V_{k+1}^s - X_{k+1}), \quad (27)$$

終端条件は  $\hat{\theta}_N^s = \hat{\theta}_N$ ,  $V_N^s = V_N$ .

■実装 `exp12(data)` を用いる. 前向きで `theta_hat`, `V_all`, `X_all` を保存し, 後向きで  $k = N-1$  から 1 へ `update_kalman_smoother` を適用して `theta_smooth`, `V_smooth` を得る. 真値と平滑化推定を描画し `graphs/task12.png` に保存する. 評価は `mean((theta_true - theta_smooth)^2)`.

■結果 データ生成と実行

```
1 kalman_data <- generate_kalman_data(seed = 42, N = 100,  
2                                     a_k = 0.9, c_k = 2,  
3                                     sigma2_v = 1, sigma2_w = 1)  
4 result12 <- exp12(kalman_data)
```

出力は `mmse`: 0.1716128 であった. 図 10 に真値と平滑化推定を示す.

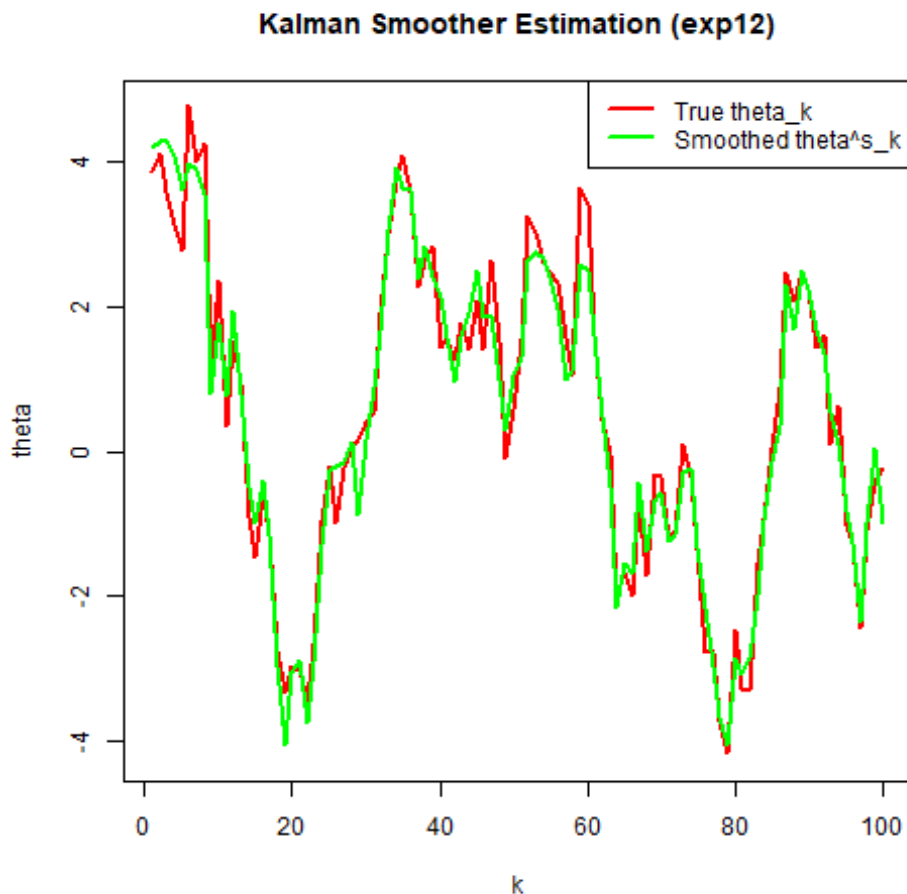


図 10: カルマン smoother の推定結果 (赤: 真値, 緑: 平滑化推定)

■考察

## 6 交互最小二乗法と K-平均法

### 6.1 原理と方法

■**交互最小二乗法** 非線形最小二乗問題のうち、パラメータを二分割すると片方ずつは線形になる場合に適用する。パラメータを  $\theta = (\alpha, \beta)^\top$  と分割し、 $g(\alpha, \beta, x)$  が  $\alpha, \beta$  に関してそれぞれ線形（双線形など）であるとする。観測  $(x_i, y_i)$  に対し、目的関数

$$J(\alpha, \beta) := \sum_{i=1}^N \|y_i - g(\alpha, \beta, x_i)\|^2 \quad (28)$$

を最小化する。片方のパラメータを固定すれば通常の線形最小二乗（式 (3.5) 型）で解ける。交互更新により  $J$  は単調非増加で、反復に伴い悪化しない。ただし一般には局所解に収束しうるため初期値が重要である。

初期値  $\alpha^{(0)}, \beta^{(0)}$  を与え、 $j = 1, 2, \dots$  について次を繰り返す。

1. **Step A ( $\beta$  固定)** :  $\beta = \beta^{(j-1)}$  として

$$\alpha^{(j)} := \arg \min_{\alpha} \sum_{i=1}^N \|y_i - g(\alpha, \beta^{(j-1)}, x_i)\|^2. \quad (29)$$

これは線形最小二乗で解く。

2. **Step B ( $\alpha$  固定)** :  $\alpha = \alpha^{(j)}$  として

$$\beta^{(j)} := \arg \min_{\beta} \sum_{i=1}^N \|y_i - g(\alpha^{(j)}, \beta, x_i)\|^2. \quad (30)$$

これも線形最小二乗で解く。

収束判定は例えば  $\|\alpha^{(j)} - \alpha^{(j-1)}\|^2 + \|\beta^{(j)} - \beta^{(j-1)}\|^2 < \varepsilon$  などとする。更新ごとに  $J(\alpha^{(j)}, \beta^{(j)}) \leq J(\alpha^{(j-1)}, \beta^{(j-1)})$  が成り立つ性質を利用する。初期値の多点スタートや正則化の併用が実務上有効である。

以下は、交互最小二乗法の R による実装例である。

Listing 4: 交互最小二乗法更新

```
1 update_alpha_beta <- function(alpha, beta, x, x_T, y, n) {
2   # Step 1: alphaを固定してbetaを推定
3   x_alpha <- matrix(0, nrow = n, ncol = 3)
4   for (i in 1:n) {
5     x_alpha[i, ] <- t(alpha) %*% x[, , i]
6   }
7   result <- regression_simple(x_alpha, y)
8   beta_new <- result$theta_hat
9
10  # Step 2: betaを固定してalphaを推定
11  x_beta <- matrix(0, nrow = n, ncol = 2)
12  for (i in 1:n) {
13    x_beta[i, ] <- t(beta_new) %*% x_T[, , i]
14  }
```

```

15 result2 <- regression_simple(x_beta, y)
16 alpha_new <- result2$theta_hat
17
18 return(list(alpha_new = alpha_new, beta_new = beta_new))
19 }

```

■K-平均法 データ  $\{x_i\}_{i=1}^N \subset \mathbb{R}^p$  を  $K$  個のクラスタに分割する. クラスタ  $V_\ell$  の代表点 (重心) を

$$\mu(V_\ell) := \arg \min_{\mu \in \mathbb{R}^p} \sum_{i \in V_\ell} \|x_i - \mu\|^2 = \frac{1}{|V_\ell|} \sum_{i \in V_\ell} x_i \quad (31)$$

と定めると, 目的は

$$\min_{\{V_\ell\}} \sum_{\ell=1}^K \sum_{i \in V_\ell} \|x_i - \mu(V_\ell)\|^2 \quad (32)$$

の最小化である. 指示変数  $r_{i,\ell} \in \{0, 1\}$  ( $x_i$  がクラスタ  $\ell$  に属すれば 1) を用いると

$$\min_{\{\mu_\ell\}, \{r_{i,\ell}\}} \sum_{\ell=1}^K \sum_{i=1}^N r_{i,\ell} \|x_i - \mu_\ell\|^2 \quad (33)$$

と書ける. 行列表示では  $X = [x_1, \dots, x_N]$ ,  $U = [\mu_1, \dots, \mu_K]$ ,  $R = (r_{i,\ell})$  として

$$\|X - UR\|_F^2 \quad (34)$$

の最小化に等価である. 大域最適性は保証されず初期値依存性が強い.

初期重心  $\{\mu_\ell^{(0)}\}_{\ell=1}^K$  を与え,  $j = 1, 2, \dots$  について次を交互に実行する.

1. 割当て (Assignment): 各データを最も近い重心へ割り当てる.

$$r_{i,\ell}^{(j)} = \begin{cases} 1 & \text{if } \ell = \arg \min_{k=1, \dots, K} \|x_i - \mu_k^{(j-1)}\|^2, \\ 0 & \text{otherwise.} \end{cases} \quad (35)$$

2. 重心更新 (Update): 各クラスタの平均で重心を更新する.

$$\mu_\ell^{(j)} = \frac{\sum_{i=1}^N r_{i,\ell}^{(j)} x_i}{\sum_{i=1}^N r_{i,\ell}^{(j)}} \quad (\sum_i r_{i,\ell}^{(j)} > 0). \quad (36)$$

停止条件は  $\max_\ell \|\mu_\ell^{(j)} - \mu_\ell^{(j-1)}\|^2 < \varepsilon$  等とする. 目的関数は反復で非増加となるが, 局所解に収束するため初期化戦略 (例: 多点スタート,  $k$ -means++) が重要である.

以下は, K-平均法の R による実装例である.

Listing 5: K-平均法

```

1 k_means <- function(x, K, max_iters = 10000, tol = 1e-6) {
2   n <- nrow(x)
3   d <- ncol(x)
4
5   # ランダムに初期クラスタ中心を選択
6   centroids <- x[sample(1:n, K), ]
7
8   cluster_assignments <- rep(0, n)

```

```

9   for (iter in 1:max_iters) {
10     # 各データポイントを最も近いクラスタ中心に割り当て
11     for (i in 1:n) {
12       distances <- apply(centroids, 1, function(centroid) sum((x[i, ] - centroid
13         )^2))
14       cluster_assignments[i] <- which.min(distances)
15     }
16     # 新しいクラスタ中心を計算
17     new_centroids <- matrix(0, nrow = K, ncol = d)
18     for (k in 1:K) {
19       points_in_cluster <- x[cluster_assignments == k, , drop = FALSE]
20       if (nrow(points_in_cluster) > 0) {
21         new_centroids[k, ] <- colMeans(points_in_cluster)
22       } else {
23         new_centroids[k, ] <- centroids[k, ] # クラスタにポイントがない場合は
24           中心を維持
25       }
26     }
27     # 収束判定
28     if (max(sqrt(rowSums((new_centroids - centroids)^2))) < tol) {
29       # 各データポイントから割り当てられた中心までの距離の合計を計算
30       total_distance <- 0
31       for (i in 1:n) {
32         centroid_idx <- cluster_assignments[i]
33         distance <- sum((x[i, ] - centroids[centroid_idx, ])^2)
34         total_distance <- total_distance + distance
35       }
36       break
37     }
38     centroids <- new_centroids
39   }
40   return(list(
41     centroids = centroids,
42     cluster_assignments = cluster_assignments,
43     total_distance = total_distance
44   ))
45 }

```

## 7 アルゴリズム比較

表や図を用いて複数手法の比較を行う。計算量・精度・安定性・実行時間の観点でまとめる。

## 8 結論

本章での結論と今後の課題を箇条書きでまとめる。

## 付録 A 使用コード一覧

主要スクリプトと入手先を列挙する。

- 実験コード (R) : <https://github.com/<your-repo>>
- データ生成スクリプト : `scripts/generate_data.jl`

## 付録 B 補足導出

必要な導出や補助的な数学的議論を載せる。

## 参考文献

## 参考文献

- [1] 数理工学実験 (2025 年度配布資料) .