数理工学実験
テーマ:最小二乗法

2回生　田中風帆 (1029321151)
実施場所:自宅

実施:2021年12月13日
提出:2021年12月26日

目 次

図 目 次

表 目 次

第 I 部
概要

このレポートでは、最小二乗法の関わる課題を解いた。重回帰問題、多項式回帰問題を解き (課題 8、9)、観測誤差の分布やデータの取り方によって結果にどのような違いが出るかを確認し、その理由を考察した (課題 10、11)。観測値の次元が多次元になる場合についても同様のことを行い (課題 12)、観測誤差の分布が異なるデータを組み合わせたものに対しても回帰を行った (課題 13)。データを分割してパラメータを推定し、それらを合成することで、全データを使用した場合と同様の推定値を得られることを確認した (課題 14)。さらに偶然誤差の分散の推定値を求め、それを用いてパラメータの推定値を合成し、全データを用いた回帰により得られたパラメータとどちらが優れているか比較、考察した (課題 15)。バネマスダンパ系を表す直線状の運動方程式のパラメータを逐次最小二乗法で求めた (課題 16)。この際、正則化項などに工夫を加えた。正弦波で変動する信号から得られたデータに対して忘却係数つきの逐次最小二乗法で回帰を行い、各時刻でパラメータを推定した (課題 17)。Kalman フィルタを実装し、与えられた離散時間ダイナミクスおよび観測方程式に適用することで時間ごとのパラメータの推定値を求めた (課題 18)。RTS アルゴリズムを実装し、課題 18 の設定のもとでパラメータの初期値を推定し、またその推定誤差分散を求めた (課題 19)。交互最小二乗法を実装し、与えられたデータに適用することでパラメータを推定した。また、初期値を変えて同様のことを繰り返し行った (課題 20)。最後に、与えられたデータに k-means 法を適用し、3 つのクラスターに分類した。異なる初期値に対して、同様のことを繰り返し行った。

第 II 部
問題 1

ここでは、N 個の入力および観測値のデータ $\{(x_i, y_i)\}(i = 1, ..., N)$ が与えられているとき、既知の関数

$$f(\theta, x) = \varphi(x)\theta \tag{0.1}$$

に関する最適化問題

$$\underset{\theta \in \mathbb{R}^n}{min} \sum_{i=1}^{N} \|y_i - f(\theta, x_i)\|^2 \tag{0.2}$$

の解が

$$\hat{\theta}_N = (\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i))^{-1} \sum_{j=1}^{N} \varphi(x_j)^T y_j \tag{0.3}$$

で求められることを示した。

1　準備

1.1　最小二乗法

次の関係式を満たす方程式が与えられているとする。

$$z = f(\theta, x)$$

ただし、x が入力、z が出力、θ がパラメータである。このとき、誤差を $w \in \mathbb{R}^m$ とおくと、観測値は以下で与えられる。

$$y = z + w \tag{1.1}$$

関数 f が既知であるがパラメータ θ が未知であり、入力と観測値のデータが与えられているとき、θ の値を決定するアルゴリズムが最小二乗法である。その具体的な式は、(0.2) で表される最適化問題であり、これを解くことで θ の値を得ることができる。

1.2　回帰問題

θ が f に対して線形であるとき、適当な行列値関数 $\mathbb{R}^p \to \mathbb{R}^{m \times n}$ が存在し、式 (0.1) が成り立つ。ここで、最適化問題 (0.2) の解は解析的に求められる。この解析解を求めるのが本問の題意である。

2　問題 1 の回答

最適化問題 (0.2) を解く。

$$F(\theta) = \sum_{i=1}^{N} \|y_i - \varphi(x_i)\theta\|^2 = \sum_{i=1}^{N} (y_i - \varphi(x_i)\theta)^T (y_i - \varphi(x_i)\theta) =$$
$$\sum_{i=1}^{N} (y_i^T - (\varphi(x_i)\theta)^T)(y_i - \varphi(x_i)\theta) = \sum_{i=1}^{N} (y_i^T - \theta^T \varphi(x_i)^T)(y_i - \varphi(x_i)\theta)$$
$$= \sum_{i=1}^{N} (y_i^T y_i - \theta^T \varphi(x_i)^T y_i - y_i^T \varphi(x_i)\theta + \theta^T \varphi(x_i)^T \varphi(x_i)\theta)$$
$$= \sum_{i=1}^{N} (y_i^T y_i - 2\theta^T \varphi(x_i)^T y_i + \theta^T \varphi(x_i)^T \varphi(x_i)\theta)$$

とおく。$\theta = \hat{\theta}_N$ がこの最適化問題の解である時、$\theta = \hat{\theta}_N$ において、$F(\theta)$ を θ で偏微分して得られる勾配ベクトルが 0 となる必要がある。$gradF(\theta) = \sum_{i=1}^{N} (-2\varphi(x_i)^T y_i + 2\varphi(x_i)^T \varphi(x_i)\theta)$ であるため、

$$\begin{aligned} gradF(\hat{\theta}_N) &= \sum_{i=1}^{N} (-2\varphi(x_i)^T y_i + 2\varphi(x_i)^T \varphi(x_i)\hat{\theta}_N) \\ &= 0 \therefore \sum_{i=1}^{N} (\varphi(x_i)^T \varphi(x_i))\hat{\theta}_N = \sum_{i=1}^{N} \varphi(x_i)^T y_i \end{aligned} \tag{2.1}$$

である。$\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i)$ が正則の時、$(\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i))^{-1}$ が存在するため、$\hat{\theta}_N = (\sum_{i=1}^{N} \varphi(x_i)^T \varphi(x_i))^{-1} \sum_{i=1}^{N} \varphi(x_i)^T y_i$ となり、式 (0.3) が導かれる。(証明終)

第 III 部
問題 2

ここでは、式

$$\hat{\theta}_{N_1+N_2} = (\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1} \times (\Phi_{Q_1,N_1}^{-1} \hat{\theta}_{N_1} + \Phi_{Q_2,N_2}^{-1} \hat{\theta}_{N_2})$$

が成立することを示した。式中、および証明中で使用されている記号については、課題 14 及び 15 の準備の項目で説明してあるため、ここでは問題 2 の回答のみを記載した。

3　準備

課題 14、15 にて詳説してあるため、ここでは省略する。

4　問題 2 の回答

$\hat{\theta}_{N_1+N_2}$ について、以下の等式が成り立つ。

$$\begin{aligned}
&\hat{\theta}_{N_1+N_2} \\
&= (\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1}(\sum_{j=1}^{N_1} \varphi_j^T Q_1 y_j) + \\
&(\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1}(\sum_{j=N_1+1}^{N_1+N_2} \varphi_j^T Q_2 y_j) \\
&= (\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1}\Phi_{Q_1,N_1}^{-1}\Phi_{Q_1,N_1}(\sum_{j=1}^{N_1} \varphi_j^T Q_1 y_j) + \\
&(\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1}\Phi_{Q_2,N_2}^{-1}\Phi_{Q_2,N_2}(\sum_{j=N_1+1}^{N_1+N_2} \varphi_j^T Q_2 y_j) \\
&= (\Phi_{Q_1,N_1}^{-1} + \Phi_{Q_2,N_2}^{-1})^{-1}(\Phi_{Q_1,N_1}^{-1}\hat{\theta}_{N_1} + \Phi_{Q_2,N_2}^{-1}\hat{\theta}_{N_2})
\end{aligned} \tag{4.1}$$

以上より示された。(証明終)

第 IV 部

課題 8

各入力 $x_i \in \mathbb{R}^2, i = 1, ..., 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = x_i^T \theta + w_i$$

ただし、観測誤差は $\mathcal{N}(0,1)$ にしたがって発生しており、既知なのは平均の情報のみであるとする。本問ではこの観測データを用いて、1 で θ の最小二乗誤差推定量と推定誤差共分散行列を求めた。2 では用いるデータを増やした時、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認した。3 では、全てのデータを用いた時の決定変数を求めた。
なお、本レポートにおいて $\mathcal{N}(a, b)$ は平均 a、分散 b の正規分布であるとする。

5　準備

式 (0.3) で表される $\hat{\theta}_N$ を最小二乗誤差推定値という。また、行列値関数が $\varphi : \mathbb{R}^p \to \mathbb{R}^{m \times n}$ であるとき、このときの観測誤差の共分散行列の推定値は

$$\hat{V_N} = \frac{1}{N-n}\sum_{i=1}^{N}(y_i - \varphi(x_i)\hat{\theta}_N)(y_i - \varphi(x_i)\hat{\theta}_N)^T \tag{5.1}$$

で与えられる。本レポート内にて、簡単のため以降は

$$\Phi_N = (\sum_{i=1}^{N} \varphi_i^T \varphi_i)^{-1}, \quad \varphi_i = \varphi(x_i) \tag{5.2}$$

を用いる。もし観測誤差の共分散行列 V が既知であるならば、推定誤差の共分散行列は

$$\Phi_N \sum_{i=1}^{N} \varphi_i^T V \varphi_i \Phi_N \tag{5.3}$$

で与えられる。観測誤差の推定値がわからない場合は、式 (5.3) の V を $\hat{V}_N$ で置き換えれば良い。
学習データがどの程度で $\hat{\theta}_N$ が収束するかを確認するには、横軸データ数、縦軸 $\hat{\theta}_N$ のグラフを表示すればよい。これは交差検証において、検証用のデータの数を確保するのに有用である。
観測誤差の確率分布が不明である場合、得られたパラメータがどれほどよくデータを説明するかの指標として決定変数が用いられる。決定変数は以下の式で表される。

$$C = \frac{\sum_{i=i}^{N} \|\varphi_i \hat{\theta}_N - \bar{y}\|^2}{\sum_{i=1}^{N} \|y_i - \bar{y}\|^2}, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{5.4}$$

決定変数は [0,1] の範囲に値をとり、その値が 1 に近いほど良い推定であるとみなされる。

6 課題 8 の回答

6.1 1.

与えられた全てのデータ $\{(x_i, y_i)\} i = 1, ..., 10000$ を用いて、θ の最小二乗誤差推定量と推定誤差共分散行列を求めた。最小二乗誤差推定量は式 (0.3) で、推定誤差共分散行列は式 (5.3) で求まることから、これらを実装した。ここで、x は二次元より $n = 2$ である。ただし、今回観測誤差の分散などが未知であることから、は式 (5.3) においては V の代わりに $\hat{V}_N$ を用いた。実装には Python のライブラリである numpy を用いた。結果は以下である。

表 1: 上から θ の最小二乗誤差推定量、推定誤差共分散行列となっている.

θ の最小二乗誤差推定量	(1.50655081 1.99769567)
推定誤差共分散行列	$\begin{pmatrix} 9.86649107e-05 & -4.08165714e-07 \\ -4.08165714e-07 & 1.00524760e-04 \end{pmatrix}$

6.2 2.

ここでは、用いるデータ数を増やした時、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認した。どのデータ数においても、データは先頭から順に採用した。ただし、$N = 2, 4, 8, ..., 2^{13}$ の時をプロットした。結果は以下である。

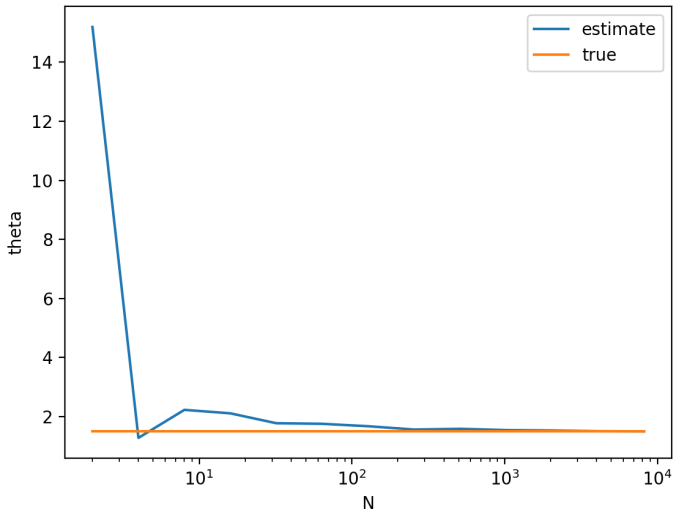

図 1: $\hat{\theta}_N$ の第一成分が収束する様子. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

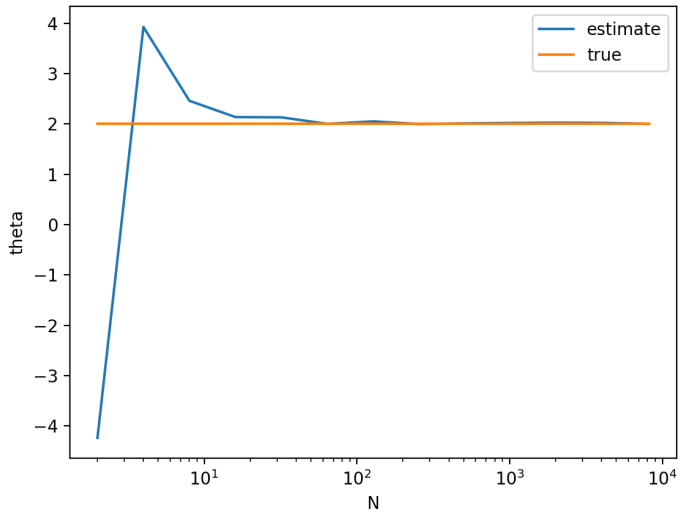

図 2: $\hat{\theta}_N$ の第二成分が収束する様子. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

6.3 3.

ここでは、式 (5.4) を実装し、全てのデータを用いた時の決定変数を求めた。結果は以下である。

表 2: 決定変数

決定変数	0.8610224003412024

7 この課題のまとめと考察

1 で求めた θ の予測値はパラメータの真の値である (1.5,2.0) とほぼ一致していた。よって、この予測の精度は良いということが予測される。1 で求めた推定誤差共分散行列の値はほぼ 0 となった。Φ_N が N に対して単調減少であり、漸近的に推定誤差が 0 に近づくことが理論的に予測されることから、この結果は正しいと言える。2 の結果から、今回の場合、観測データが 1000 個程度あればパラメータは収束するということがわかった。また、3 の決定変数より、得られたパラメータを用いて構成されるモデルは高い精度でデータを表現することがわかった。

8 コード

コード 1: 課題 8 のコード

```
1  import pandas as pd
2  import numpy as np
3  import numpy.linalg as la
4  import matplotlib.pyplot as plt
5
6  #基底関数
7  def phi(x):
8      return x
9
10 #パラメータを求める
11 def calc_theta(x, y):
12     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
13     ans_kari = np.dot(Phi, np.transpose(phi(x)))
14     ans = np.dot(ans_kari, y)
15     return ans
16
17 #データを取る
18 df = pd.read_csv("./suri_jikken6_data/mmse_kadai1.txt",header=
       None)
19 data = np.array(df)
20 n = 2
21 N = 10000
22
23 x = np.array(data[:,0:2])
24 y = np.array(data[:,2])
25
26 #全てのデータを用いてパラメータを予測
27 pred_ans = calc_theta(x, y)
28 print(pred_ans)
29
30 #推定誤差共分散行列を求める
31 V_n_hat_kari = y − np.dot(phi(x), pred_ans)
32 V_n_hat_kari = np.dot(V_n_hat_kari, np.transpose(V_n_hat_kari))
33 V_n_hat = V_n_hat_kari / (N − n)
34 Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
35 Err_mat = Phi.dot(V_n_hat ∗ x.T.dot(x)).dot(Phi)
36 print(Err_mat)
37
38 #プロットのためのリスト
39 theta_0_list = []
40 theta_1_list = []
41 N_list = []
42 real_ans_0_list = []
43 real_ans_1_list = []
44
45 #データの数を変えてパラメータを求める
46 for i in range(1,14):
47     data_num = 2∗∗i
48     N_list.append(data_num)
49     real_ans_0_list.append(1.5)
50     real_ans_1_list.append(2.0)
51
52     x = np.array(data[0:data_num,0:2])
53     y = np.array(data[0:data_num,2])
54     y = np.reshape(y, (data_num,1))
55     theta_0 = calc_theta(x, y)[0]
56     theta_1 = calc_theta(x, y)[1]
57     theta_0_list.append(theta_0)
```

```
58      theta_1_list.append(theta_1)
59
60  #プロット
61  fig, ax = plt.subplots(facecolor="w")
62  ax.plot(N_list, theta_0_list, label="estimate")
63  ax.plot(N_list, real_ans_0_list, label="true")
64  ax.legend()
65
66  plt.xscale('log')
67  plt.xlabel('N')
68  plt.ylabel('theta')
69  plt.show()
70
71  fig, ax = plt.subplots(facecolor="w")
72  ax.plot(N_list, theta_1_list, label="estimate")
73  ax.plot(N_list, real_ans_1_list, label="true")
74  ax.legend()
75
76  plt.xscale('log')
77  plt.xlabel('N')
78  plt.ylabel('theta')
79  plt.show()
80
81  #決定変数を求める
82  y_bar = np.mean(y)
83  bunsi = np.sum((np.dot(phi(x),pred_ans) - y_bar)**2)
84  bunbo = np.sum((y - y_bar)**2)
85  C = bunsi / bunbo
86  print(C)
```

第 V 部

課題 9

各入力 $x_i \in \mathbb{R}, i = 1, ..., 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = \varphi(x_i)\theta + w_i, \quad \varphi(x) = [1 \quad x \quad x^2 \quad x^3]$$

ただし、観測誤差は $\mathcal{N}(0,9)$ にしたがって発生しており、既知なのは平均の情報のみであるとする。本問ではこの観測データを用いて、1 で θ の最小二乗誤差推定量と推定誤差共分散行列を求めた。2 では用いるデータを増やした時、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認した。3 では、全てのデータを用いた時の決定変数を求めた。

9　準備

この課題を解くのに必要な前提知識は課題 8 にてまとめているため、ここでは省略する。

10　課題 9 の回答

10.1　1.

与えられた全てのデータ $\{(x_i, y_i)\}i = 1, ..., 10000$ を用いて、θ の最小二乗誤差推定量と推定誤差共分散行列を求めた。最小二乗誤差推定量は式 (0.3) で、推定誤差共分散行列は式 (5.3) で求まることから、これらを実装した。ここで、φ は 4 次元であるため、$n = 4$ である。ただし、今回観測誤差の分散などが未知であることから、は式 (5.3) においては V の代わりに $\hat{V}_N$ を用いた。実装には Python のライブラリである numpy を用いた。結果は以下である。

表 3: 上から θ の最小二乗誤差推定量、推定誤差共分散行列となっている.

θ の最小二乗誤差推定量
(-0.50902942 1.97586067 0.19774405 -0.09866691)
推定誤差共分散行列
$\begin{pmatrix} 2.02344200e-03 & -1.18649745e-05 & -1.34831217e-04 & 3.97116548e-07 \\ -1.18649745e-05 & 6.75301451e-04 & -1.89854517e-07 & -3.79471936e-05 \\ -1.34831217e-04 & -1.89854517e-07 & 1.60391021e-05 & 6.35182005e-08 \\ 3.97116548e-07 & -3.79471936e-05 & 6.35182005e-08 & 2.52871068e-06 \end{pmatrix}$

10.2　2.

ここでは、用いるデータ数を増やした時、$\hat{\theta}_N$ の各要素が収束していくことを片対数グラフでプロットして確認した。どのデータ数においても、データは先頭から順に採用した。ただし、$N = 4, 8, ..., 2^{13}$ の時をプロットした。結果は以下である。

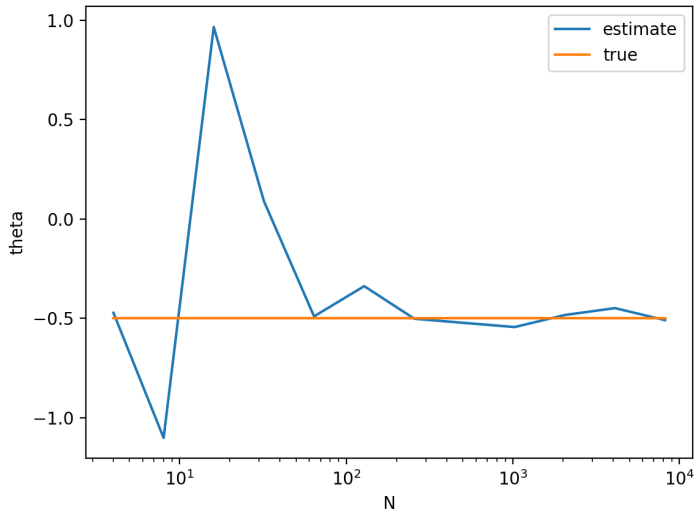

図 3: $\hat{\theta}_N$ の第一成分が収束する様子. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

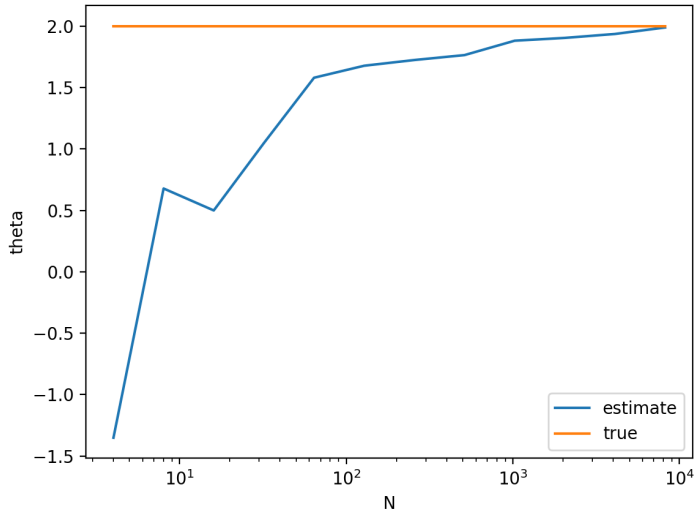

図 4: $\hat{\theta}_N$ の第二成分が収束する様子. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

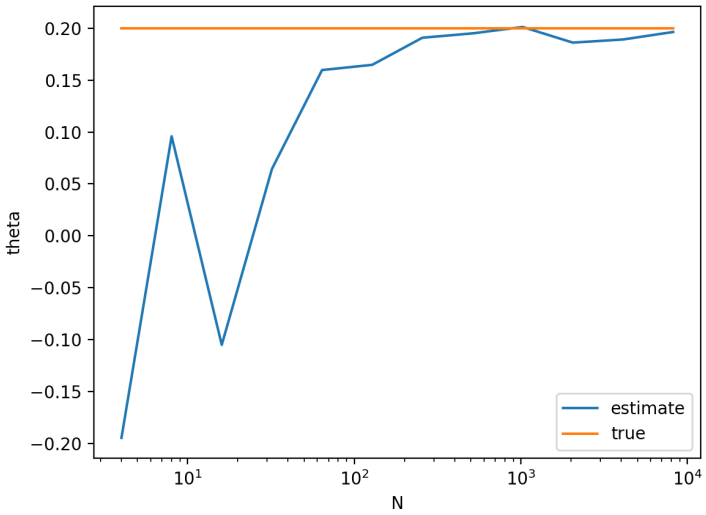

図 5: $\hat{\theta}_N$ の第三成分が収束する様子. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第三成分、横軸が用いたデータ数.(ただし log スケールである)

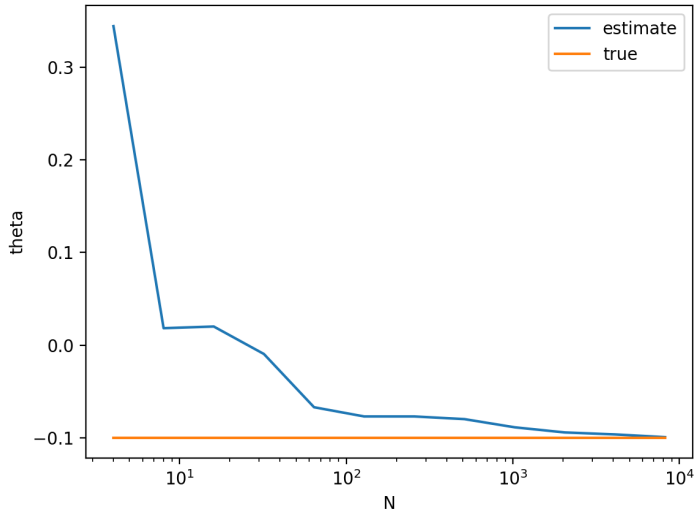

図 6: $\hat{\theta}_N$ の第四成分が収束する様子. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第四成分、横軸が用いたデータ数.(ただし log スケールである)

10.3 3.

ここでは、式 (5.4) を実装し、全てのデータを用いた時の決定変数を求めた。結果は以下である。

表 4: 決定変数

決定変数	0.462430059307111

11 この課題のまとめと考察

パラメータの真値が (-0.5,2,0.2,-0.1) であることから、もとまったパラメータは真の値に近いことがわかった。
1 で求めた推定誤差共分散行列の値はほぼ 0 となった。Φ_N が N に対して単調減少であり、漸近的に推定誤差が 0 に近づくことが理論的に予測されることから、この結果は正しいと言える。2 の結果から、今回の場合、観測データが 10000 個、あるいはそれ以上に至るまでパラメータは収束しないということがわかった。また、3 の決定変数より、得られたパラメータを用いて構成されるモデルはあまり良い推定ではないことがわかった。

12 コード

コード 2: 課題 9 のコード

```
1  import pandas as pd
2  import numpy as np
3  import numpy.linalg as la
4  import matplotlib.pyplot as plt
5
6  #基底関数
7  def phi(x):
8      kitei = np.array([x**0, x, x**2, x**3])
9      return np.transpose(kitei)
10
11 #パラメータの予測値を計算
12 def calc_theta(x, y):
13     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
14     ans_kari = np.dot(Phi, np.transpose(phi(x)))
15     ans = np.dot(ans_kari, y)
16     return ans
17
18 #データ読み込み
19 df = pd.read_csv("./suri_jikken6_data/mmse_kadai2.txt",header=
       None)
20 data = np.array(df)
21 n = 4
22 N = 10000
23
```

```
x = np.array(data[:,0])
y = np.array(data[:,1])

#パラメータ計算
pred_ans = calc_theta(x, y)
print(pred_ans)

#推定誤差共分散行列を求める
V_n_hat_kari = y − np.dot(phi(x), pred_ans)
V_n_hat_kari = np.dot(V_n_hat_kari, np.transpose(V_n_hat_kari))
V_n_hat = V_n_hat_kari / (N − n)
Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
Err_mat = Phi.dot(V_n_hat ∗ phi(x).T.dot(phi(x))).dot(Phi)
print(Err_mat)

#プロットのためのリスト
theta_0_list = []
theta_1_list = []
theta_2_list = []
theta_3_list = []
N_list = []
real_ans_0_list = []
real_ans_1_list = []
real_ans_2_list = []
real_ans_3_list = []

#データ数を変えてパラメータ推定
for i in range(2,14):
    data_num = 2**i
    N_list.append(data_num)
    real_ans_0_list.append(−0.5)
    real_ans_1_list.append(2.0)
    real_ans_2_list.append(0.2)
    real_ans_3_list.append(−0.1)

    x = np.array(data[0:data_num,0])
    y = np.array(data[0:data_num,1])

    theta_0 = calc_theta(x, y)[0]
    theta_1 = calc_theta(x, y)[1]
    theta_2 = calc_theta(x, y)[2]
    theta_3 = calc_theta(x, y)[3]
    theta_0_list.append(theta_0)
    theta_1_list.append(theta_1)
    theta_2_list.append(theta_2)
    theta_3_list.append(theta_3)

#プロット
fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_0_list, label="estimate")
ax.plot(N_list, real_ans_0_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_1_list, label="estimate")
ax.plot(N_list, real_ans_1_list, label="true")
ax.legend()
```

```
plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_2_list, label="estimate")
ax.plot(N_list, real_ans_2_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_3_list, label="estimate")
ax.plot(N_list, real_ans_3_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

#決定変数を求める
y_bar = np.mean(y)
bunsi = np.sum((np.dot(phi(x),pred_ans) - y_bar)**2)
bunbo = np.sum((y - y_bar)**2)
C = bunsi / bunbo
print(C)

#決定変数 0.4624300593071111
#[-0.50902942 1.97586067 0.19774405 -0.09866691]
#[[ 2.02344200e-03 -1.18649745e-05 -1.34831217e-04 3.97116548e
    -07]
#[-1.18649745e-05 6.75301451e-04 -1.89854517e-07 -3.79471936e
    -05]
#[-1.34831217e-04 -1.89854517e-07 1.60391021e-05 6.35182005e
    -08]
#[ 3.97116548e-07 -3.79471936e-05 6.35182005e-08 2.52871068e-06]]
```

第 VI 部

課題 10

ここでは、各入力 $x_i \in \mathbb{R}^2, i = 1, ..., 10000$ に対し、次の式で観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = x_i^T \theta + w_i$$

ただし、観測誤差は標準 Cauchy 分布に従うとする。本問ではこの観測データを用いて、1 で θ の最小二乗誤差推定量を、式 (0.3) を用いて求めた。2 では用いるデータを増やした時、$\hat{\theta}_N$ の各要素が収束して行かないことを片対数グラフでプロットして確認した。

13　準備

この課題を解くのに必要な前提知識は課題 8 にて述べているため省略する。

14　課題 10 の回答

与えられた全てのデータ $\{(x_i, y_i)\}i = 1, ..., 10000$ を用いて、θ の最小二乗誤差推定量を求めた。最小二乗誤差推定量は式 (0.3) で求まることから、これらを実装した。実装には Python のライブラリである numpy を用いた。結果は以下である。

表 5: θ の最小二乗誤差推定量

θ の最小二乗誤差推定量
(2.3730741 1.53731124)

14.1　2.

ここでは、用いるデータ数を増やした時、$\hat{\theta}_N$ の各要素が収束していかないことを片対数グラフでプロットして確認した。どのデータ数においても、データは先頭から順に採用した。ただし、$N = 4, 8, ..., 2^{13}$ の時をプロットした。結果は以下である。

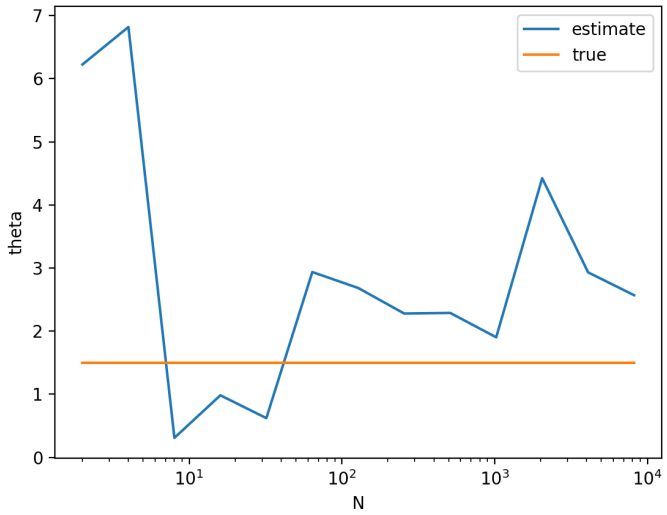

図 7: $\hat{\theta}_N$ の第一成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

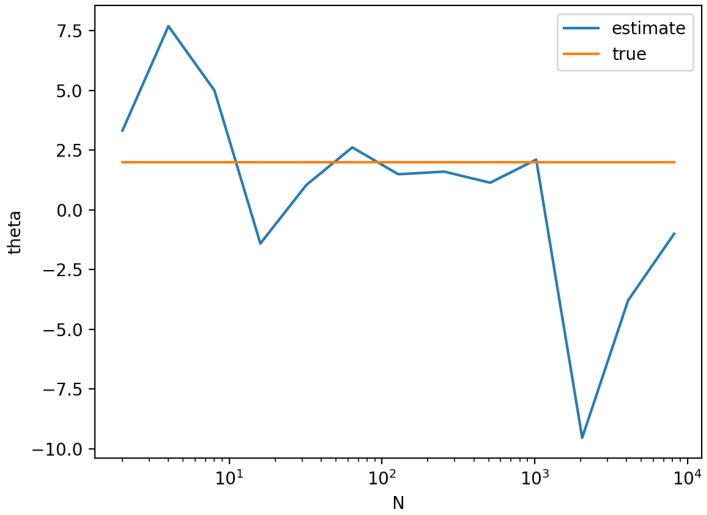

図 8: $\hat{\theta}_N$ の第二成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

15　この課題のまとめと考察

2 のプロットの結果から、最小二乗法による推定では $\hat{\theta}_N$ が収束しないことが確認できた。これは観測誤差が分散 ∞ の確率分布にしたがっていることに起因すると考えられる。このようなデータに対しては最小二乗法とは別の推定手法を適用する必要があると結論づけることができる。

16　コード

コード 3: 課題 10 のコード

```
1  import pandas as pd
2  import numpy as np
3  import numpy.linalg as la
4  import matplotlib.pyplot as plt
5
6  #基底関数
7  def phi(x):
8      return x
9
10 #パラメータ推定の関数
11 def calc_theta(x, y):
12     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
13     ans_kari = np.dot(Phi, np.transpose(phi(x)))
14     ans = np.dot(ans_kari, y)
15     return ans
16
17 #データ読み込み
18 df = pd.read_csv("./suri_jikken6_data/mmse_kadai3.txt",header=
       None)
19 data = np.array(df)
20 n = 2
21 N = 10000
22
23 x = np.array(data[:,0:2])
24 y = np.array(data[:,2])
25 y = np.reshape(y, (N,1))
26
27 #パラメータを計算
28 pred_ans = calc_theta(x, y)
29 print(pred_ans)
30 theta_0_list = []
31 theta_1_list = []
32 N_list = []
33 real_ans_0_list = []
34 real_ans_1_list = []
35
36 #データ数を変えてパラメータを推定
37 for i in range(1,14):
38     data_num = 2**i
39     N_list.append(data_num)
40     real_ans_0_list.append(1.5)
41     real_ans_1_list.append(2.0)
42
43     x = np.array(data[0:data_num,0:2])
44     y = np.array(data[0:data_num,2])
45
```

```
46      y = np.reshape(y, (data_num,1))
47      theta_0 = calc_theta(x, y)[0][0]
48      theta_1 = calc_theta(x, y)[1][0]
49      theta_0_list.append(theta_0)
50      theta_1_list.append(theta_1)
51
52  #プロット
53  fig, ax = plt.subplots(facecolor="w")
54  ax.plot(N_list, theta_0_list, label="estimate")
55  ax.plot(N_list, real_ans_0_list, label="true")
56  ax.legend()
57
58  plt.xscale('log')
59  plt.xlabel('N')
60  plt.ylabel('theta')
61  #グラフの名前
62  plt.show()
63
64  fig, ax = plt.subplots(facecolor="w")
65  ax.plot(N_list, theta_1_list, label="estimate")
66  ax.plot(N_list, real_ans_1_list, label="true")
67  ax.legend()
68
69  plt.xscale('log')
70  plt.xlabel('N')
71  plt.ylabel('theta')
72  #グラフの名前
73  plt.show()
74
75  #[[2.3730741 ][1.53731124]]]
```

第 VII 部

課題 11

ここでは、課題 9 と同じ $\varphi(x)$、θ および観測誤差の分布 $\mathcal{N}(0,9)$ を考えた。また、$x_i \in [0,1], i = 1, ..., 1000$ の時にデータ $\{(x_i, y_i)\}, i = 1, ..., 1000$ が与えられているものとする時、$\hat{\theta}_{N=1000}$ を求め、課題 9.1 の $N = 10000$ の結果と比較した。また、この結果を受けてどのようにデータを取るべきか考察した。

17　準備

この課題を解くのに必要な前提知識はすでに課題 8 で述べているため省略する。

18　課題 11 の回答

与えられた全てのデータ $\{(x_i, y_i)\}i = 1, ..., 1000$ を用いて、θ の最小二乗誤差推定量を求めた。最小二乗誤差推定量は式 (0.3) で求まることから、これら

を実装した。ただし、今回観測誤差の分散などが未知であることから、は式 (5.3) においては V の代わりに $\hat{V}_N$ を用いた。実装には Python のライブラリである numpy を用いた。結果は以下である。これと 9 で 10000 個のデータ

表 6: θ の最小二乗誤差推定量

θ の最小二乗誤差推定量
(-0.5777248 2.09682276 -0.39504672 0.54912487)

を用いて求めた (-0.50902942 1.97586067 0.19774405 -0.09866691) とを比較する。パラメータの真値が (-0.5,2.0,0.2,-0.1) であることから、パラメータのどの要素においても 9 の結果が 11 の結果よりも正確な値となっていることがわかった。用いたデータ数以外の条件は 9 と 11 で変化がないことから、この精度の差はデータ数に起因すると結論づけることができる。
9.2 のプロットから分かるように、この観測データは 10000 個またはそれ以上のデータを用いて学習を行わなければパラメータが収束しない。よって、本問であつかったデータ数 (1000 個) ではパラメータが収束しきっておらず、精度が低くなったのだと思われる。
機械学習を実際の問題解決に利用しようとする際、データの収集が困難であったり、収集にコストがかかる場合も多数あることが予想される。よって、ある程度の量のデータが集まった時点でまずそのデータを用いて 9.2 のようなプロットを行い、データの増加とともにパラメータの変化が見られるか、あるいは早い段階で収束しているかを観察する。課題 8 のように 1000 個程度のデータ数でパラメータの収束が見られる場合もあれば、本問のようにまだ収束しない場合もあるためである。変化が見られるならばパラメータが収束するまで追加でデータを集め、すでに収束しているならばデータの収集を切り上げる。切り上げる場合、精度が出ているならば終了、出ていないならば基底関数の吟味などに労力をさくのが得策ではないかと推察する。
あるいは、パラメータの次元が多い場合などは、グラフの縦軸をパラメータの代わりに決定変数としてプロットすることも考えられる。この場合、プロットを見るだけでパラメータの収束だけでなく精度の良し悪しもわかるというメリットがある一方で、決定変数を何度も計算しなければならないという計算コストの面でのデメリットがあると思われる。

19 コード

コード 4: 課題 11 のコード

```
1 import pandas as pd
2 import numpy as np
3 import numpy.linalg as la
```

```
4  import matplotlib.pyplot as plt
5
6  #基底関数
7  def phi(x):
8      kitei = np.array([x**0, x, x**2, x**3])
9      return np.transpose(kitei)
10
11 #パラメータを推定する関数
12 def calc_theta(x, y):
13     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
14     ans_kari = np.dot(Phi, np.transpose(phi(x)))
15     ans = np.dot(ans_kari, y)
16     return ans
17
18 #データ読み込み
19 df = pd.read_csv("./suri_jikken6_data/mmse_kadai4.txt",header=
       None)
20 data = np.array(df)
21 n = 3
22 N = 1000
23
24 x = np.array(data[:,0])
25 y = np.array(data[:,1])
26
27 #パラメータを推定
28 pred_ans = calc_theta(x, y)
29 print(pred_ans)
```

第 VIII 部

課題 12

ここでは、観測値の次元が 2 次元となる場合の課題を解いた。
$x_i \in \mathbb{R}, i = 1, ..., 1000$ に対し、観測値が

$$y_i = \varphi x_i \theta + w_i \tag{19.1}$$

で与えられる。ただし、w_i は $\mathcal{N}(\prime, \infty)$ の独立同一分布に従い、

$$\varphi = \begin{pmatrix} 1 & x \\ 1 & x^2 \end{pmatrix}, V = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix} \tag{19.2}$$

とする。この時、式 (0.3) および式

$$\hat{\theta}_N = (\sum_{i=1}^{N} \varphi_i^T Q_i \varphi_i)^{-1} \sum_{j=1}^{N} \varphi_j^T Q_j y_j \tag{19.3}$$

(ただし Q は重み行列) を用いてパラメータの推定値を求め、それぞれの推定誤差共分散を求めた。今回、$Q = V^{-1}$ とした。また、$\hat{\theta}_N$ の各要素の収束の様子をプロットした。

20　準備

この課題を解くのに必要な前提知識および理論を述べる。
データを得た時、それぞれのデータの信用度を考えて最小二乗法を適用したい場合がある。この時、重み行列 $Q_i, i = 1, ..., N$(ただし Q は半正定値対称行列) を用いた重み付き最小二乗法

$$\underset{\theta \in \mathbb{R}^n}{min} \sum_{i=1}^{N} (y_i - \varphi_i \theta)^T Q_i (y_i - \varphi_i \theta) \tag{20.1}$$

を考えれば良い。この時、$\sum_{i=1}^{N} \varphi_i^T Q_i \varphi_i$ が正則ならば、最適な推定値は式 (19.3) で表されることとなる。V は正定値対称行列であるため、$V = W^2$ となる正定値行列 W が一意に存在し、V の平方根行列の三乗となる。これを用いると、式 (1.1) は

$$W^{-1} y = W^{-1} \varphi(x) \theta + W^{-1} w \tag{20.2}$$

と書き直せる。ここで、$W^{-1}w$ は単位行列を分散として持つ観測誤差である。この解は

$$\hat{\theta}_N = (\sum_{i=1}^{N} \varphi_i^T W^{-2} \varphi_i)^{-1} \sum_{j=1}^{N} \varphi_j^T W^{-2} y_j \tag{20.3}$$

となるため、$Q_i = aW^{-2} = aV^{-1} (a \in \mathbb{R}$ は正定数) と選んだものに相当する。この規則により求められる推定量は最も推定誤差と訓練誤差の分散を小さくすることが理論的に知られている。
また、式 (19.3) を用いて導かれるパラメータの推定誤差共分散行列は、V を観測誤差の共分散行列として以下で表される。

$$\begin{aligned} &\Phi' \sum_{i=1}^{N} \varphi_i^T Q V Q^T \varphi_i \Phi'^T \\ &\Phi' = (\sum_{i=1}^{N} \varphi_i^T Q \varphi_i)^{-1} \end{aligned} \tag{20.4}$$

導出は以下。
$\Phi' = (\sum_{i=1}^{N} \varphi_i^T Q \varphi_i)^{-1}$ とする。この時、

$$\theta - \hat{\theta}_N = \Phi' (\Phi'^{-1} \theta - \sum_{j=1}^{N} \varphi_j Q_j y_j) = \Phi'^T \sum_{i=1}^{N} (\varphi_i^T Q w_i)$$

$$(\theta - \hat{\theta}_N)^T = \sum_{i=1}^{N} (w_i^T Q^T \varphi_i) \Phi'^T$$

よって、

$$E[(\theta - \hat{\theta}_N)(\theta - \hat{\theta}_N)^T] = E[\Phi' \sum_{i=1}^{N} (\varphi_i^T Q w_i) \sum_{j=1}^{N} (w_j^T Q^T \varphi_j) \Phi'^T] = \Phi' \sum_{i=1}^{N} \varphi_i^T Q V Q^T \varphi_i \Phi'^T$$

以上より示された。

21　課題 12 の回答

21.1　式 (0.3) を用いた場合

推定値を式 (0.3) で、推定誤差分散を式式 (5.3) と与えられた V で求めた。結果、推定値と推定誤差共分散は以下のようになった。$\hat{\theta}_N$ の各要素の収束の

表 7: 上から順にパラメータの推定値、推定誤差分散

パラメータの推定量
(2.99456713 -2.06897079)
推定誤差共分散
$\begin{pmatrix} 0.03514546 & -0.01251624 \\ -0.01251624 & 0.01086049 \end{pmatrix}$

仕方をプロットしたものは以下である。ただし、$N = 1, ..., 1000$ の場合をプロットした。

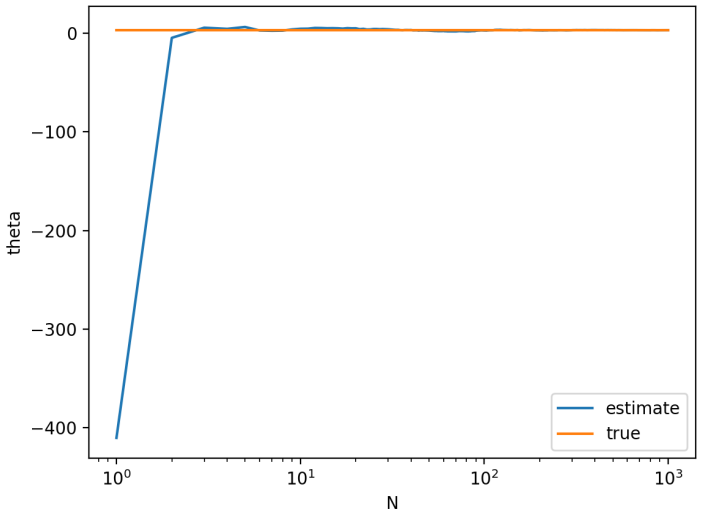

図 9: $\hat{\theta}_N$ の第一成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

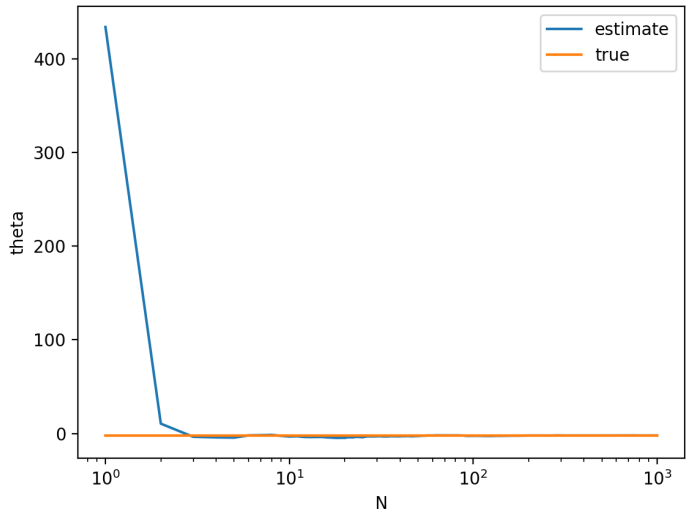

図 10: $\hat{\theta}_N$ の第二成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

21.2 式 (19.3) を用いた場合

推定値を式 (19.3) で、推定誤差分散を式 (20.4) と与えられた V、および $Q_i = V^{-1}$ で求めた。結果、推定値と推定誤差共分散は以下のようになった。$\hat{\theta}_N$ の各要素の収束の仕方をプロットしたものは以下である。ただし、

表 8: 上から順にパラメータの推定値、推定誤差分散

パラメータの推定量
(2.93908407 -1.98646467)
推定誤差共分散
$\begin{pmatrix} 0.00147742 & -0.00050005 \\ -0.00050005 & 0.00051312 \end{pmatrix}$

$N = 1, ..., 1000$ の場合をプロットした。

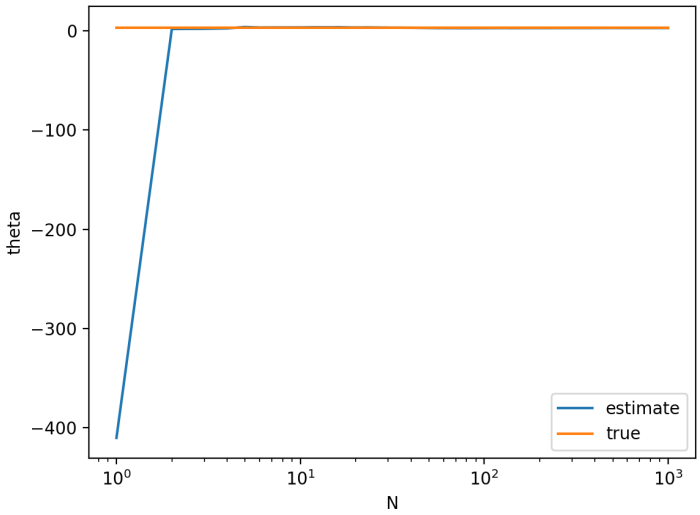

図 11: $\hat{\theta}_N$ の第一成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

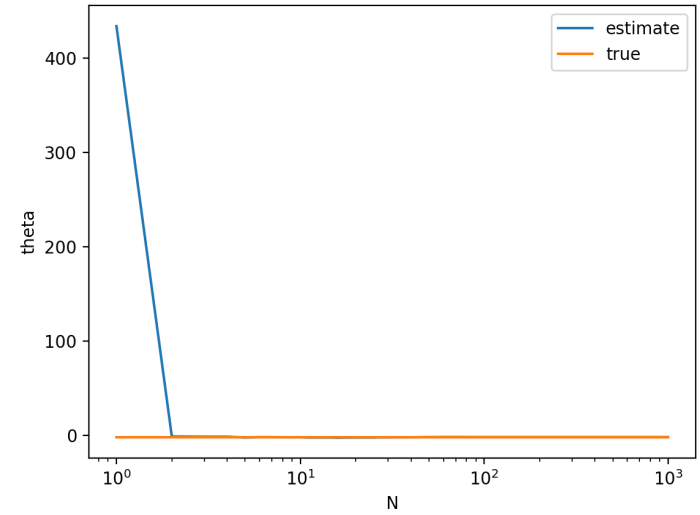

図 12: $\hat{\theta}_N$ の第二成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

22　この課題のまとめと考察

式 (19.3) を用いた場合の推定誤差共分散の方が式 (0.3) を用いた場合の推定誤差分散より 0 行列に近くなることから、式 (19.3) を用いて求めたパラメータの方が精度が良いことがわかった。また、プロットから、式 (19.3) を用いた場合の方がパラメータの収束に必要なデータ数が少ないことがわかった。準備の項目で述べたように、式 (19.3) を用いて求めた推定量は最も推定誤差の分散を小さくすることが理論的にわかっているため、本問の実験結果は妥当であると言える。

23　コード

コード 5: 課題 12 のコード

```
import pandas as pd
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt

n = 2
N = 1000
V = np.array([[100.0, 0.0],[0.0, 1.0]])

#基底関数
def phi(x):
    kitei = np.array([[x**0, x**0],[x, x**2]])
    return kitei.T

#6_5 の方法でパラメータを予測、誤差共分散行列を求める
def calc_theta_6_5(x, y,data_num):
    #パラメータ
    Phi = np.array([[0.0,0.0],[0.0,0.0]])
    for i in range(data_num):
        Phi += np.dot(phi(x)[i].T, phi(x)[i])
    Phi = la.inv(Phi)

    migi = np.array([0.0,0.0])
    for i in range(data_num):
        migi += np.dot(phi(x)[i].T, y[i])

    theta = np.dot(Phi, migi)

    #誤差共分散行列
    Err_mat = np.array([[0.0,0.0],[0.0,0.0]])

    for i in range(data_num):
        phi_v = np.dot(phi(x)[i].T, V)
        phi_v_phi = np.dot(phi_v, phi(x)[i])
        phi_v_phi_Phi = np.dot(phi_v_phi, Phi)
        Err_mat += phi_v_phi_Phi
    Err_mat = np.dot(Phi, Err_mat)

    return theta, Err_mat

#6の方法でパラメータを予測、誤差共分散行列を求める_17
def calc_theta_6_17(x, y, data_num):
```

```
43      #パラメータ
44      Q = la.inv(V)
45      Phi = np.array([[0.0,0.0],[0.0,0.0]])
46      for i in range(data_num):
47          phi_q = np.dot(phi(x)[i].T, Q)
48          Phi += np.dot(phi_q, phi(x)[i])
49      Phi = la.inv(Phi)
50      migi = np.array([0.0,0.0])
51      for i in range(data_num):
52          phi_q = np.dot(phi(x)[i].T, Q)
53          migi += np.dot(phi_q, y[i])
54
55      theta = np.dot(Phi, migi)
56
57      #誤差共分散行列
58      Err_mat = np.array([[0.0,0.0],[0.0,0.0]])
59      Phi_dash = np.array([[0.0, 0.0],[0.0, 0.0]])
60      for i in range(data_num):
61          phi_q = np.dot(phi(x)[i].T, Q)
62          phiqphi = np.dot(phi_q, phi(x)[i])
63          Phi_dash += phiqphi
64      Phi_dash = la.inv(Phi_dash)
65      for i in range(data_num):
66          kari = np.dot(phi(x)[i].T, Q)
67          kari = np.dot(kari, V)
68          kari = np.dot(kari, Q.T)
69          kari = np.dot(kari, phi(x)[i])
70          kari = np.dot(kari, Phi_dash.T)
71          Err_mat += kari
72      Err_mat = np.dot(Phi_dash ,Err_mat)
73      '''
74      for i in range(data_num):
75          phi_v = np.dot(phi(x)[i].T, V)
76          phi_v_phi = np.dot(phi_v, phi(x)[i])
77          phi_v_phi_Phi = np.dot(phi_v_phi, Phi)
78          Err_mat += phi_v_phi_Phi
79      Err_mat = np.dot(Phi, Err_mat)
80      '''
81      return theta, Err_mat
82
83  #データ読み込み
84  df = pd.read_csv("./suri_jikken6_data/mmse_kadai5.txt",header=
        None)
85  data = np.array(df)
86
87  x = np.array(data[:,0])
88  y = np.array(data[:,1:3])
89
90  #6_5 でパラメータを求める
91  ans_6_5 = calc_theta_6_5(x, y, N)
92  print(ans_6_5[0], ans_6_5[1])
93
94  #6でパラメータを求める_17
95  ans_6_17 = calc_theta_6_17(x, y, N)
96  print(ans_6_17[0], ans_6_17[1])
97
98  #プロットのためのリスト
99  theta_0_list = []
100 theta_1_list = []
101 theta_2_list = []
102 theta_3_list = []
103 N_list = []
104 real_ans_0_list = []
```

```
real_ans_1_list = []
real_ans_2_list = []
real_ans_3_list = []

#プロット
for data_num in range(1,N+1):
    #data_num = 2 ** k
    N_list.append(data_num)
    real_ans_0_list.append(3.0)
    real_ans_1_list.append(−2.0)
    real_ans_2_list.append(3.0)
    real_ans_3_list.append(−2.0)

    x = np.array(data[0:data_num,0])
    y = np.array(data[0:data_num,1:3])

    theta_0 = calc_theta_6_5(x, y, data_num)[0][0]
    theta_1 = calc_theta_6_5(x, y, data_num)[0][1]
    theta_2 = calc_theta_6_17(x, y, data_num)[0][0]
    theta_3 = calc_theta_6_17(x, y, data_num)[0][1]

    theta_0_list.append(theta_0)
    theta_1_list.append(theta_1)
    theta_2_list.append(theta_2)
    theta_3_list.append(theta_3)

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_0_list, label="estimate")
ax.plot(N_list, real_ans_0_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_1_list, label="estimate")
ax.plot(N_list, real_ans_1_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_2_list, label="estimate")
ax.plot(N_list, real_ans_2_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_3_list, label="estimate")
ax.plot(N_list, real_ans_3_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
```

```
168 plt.ylabel('theta')
169 plt.show()
```

第 IX 部
課題 13

$x_i \in \mathbb{R}, i = 1, ..., 1000$ に対し、観測値が

$$y_i = \varphi(x_i)\theta + w_i \tag{23.1}$$

で与えられる。ただし、w_i は $i = 1, ..., 500$ は $\mathcal{N}(0, V_1)$ の独立同一分布に、$i = 501, ..., 1000$ は $\mathcal{N}(0, V_2)$ の独立同一分布に従うとし、

$$\varphi(x) = \begin{pmatrix} 1 & x \\ 1 & x^2 \end{pmatrix}, V_1 = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix}, V_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \tag{23.2}$$

とする。この時、式 (0.3) および式 (19.3) を用いて推定値を求め、それぞれの収束の仕方を片対数グラフでプロットした。

24　準備

この課題を解くのに必要な前提知識は課題 12 において述べているため、省略する。

25　課題 13 の回答

25.1　式 (0.3) を用いた場合

推定値を式 (0.3) で求めた。結果、推定値は以下のようになった。$\hat{\theta}_N$ の各

表 9: パラメータの推定値

パラメータの推定量
(3.18835631 -2.09218316)

要素の収束の仕方をプロットしたものは以下である。ただし、$N = 1, ..., 1000$ の場合をプロットした。データは 1000 個のデータからランダムに選択した。

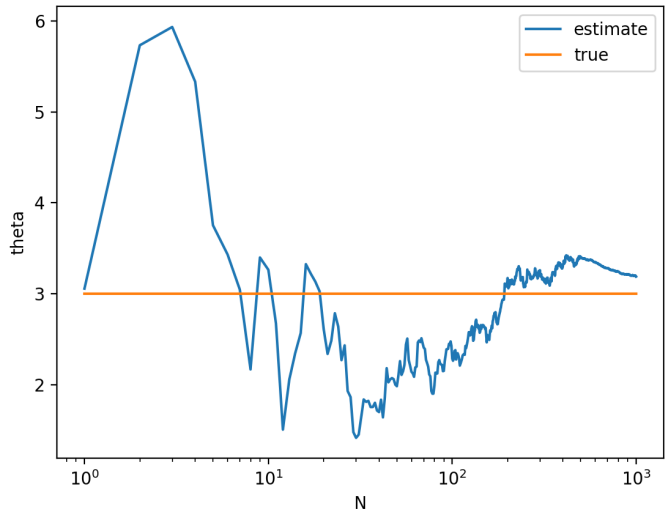

図 13: $\hat{\theta}_N$ の第一成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

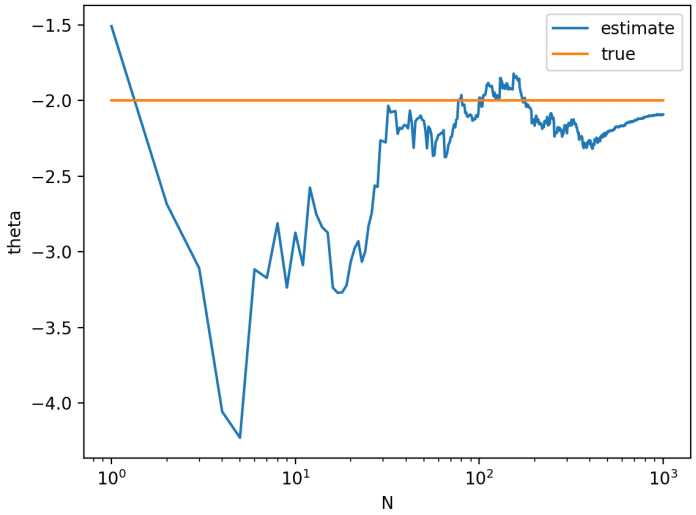

図 14: $\hat{\theta}_N$ の第二成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

25.2　式 (19.3) を用いた場合

パラメータの推定値を式 (19.3) で求めた。結果、推定値は以下のようになった。$\hat{\theta}_N$ の各要素の収束の仕方をプロットしたものは以下である。ただ

表 10: パラメータの推定値

パラメータの推定量
(2.9942022 -2.01469917)

し、$N = 1, ..., 1000$ の場合をプロットした。データは 1000 個のデータからランダムに選択した。

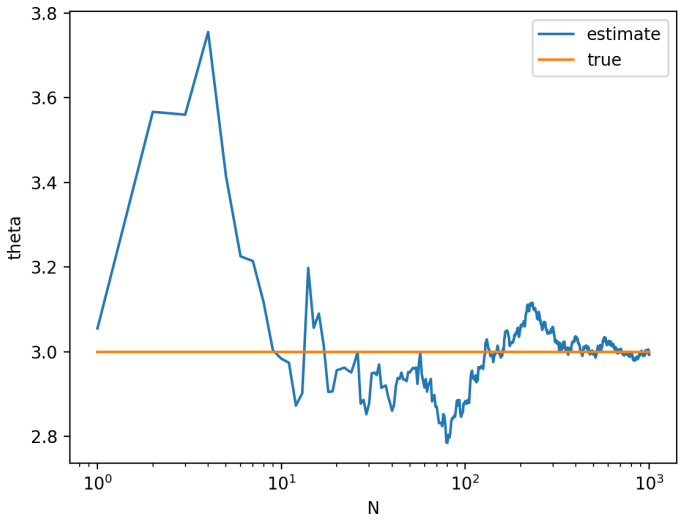

図 15: $\hat{\theta}_N$ の第一成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第一成分、横軸が用いたデータ数.(ただし log スケールである)

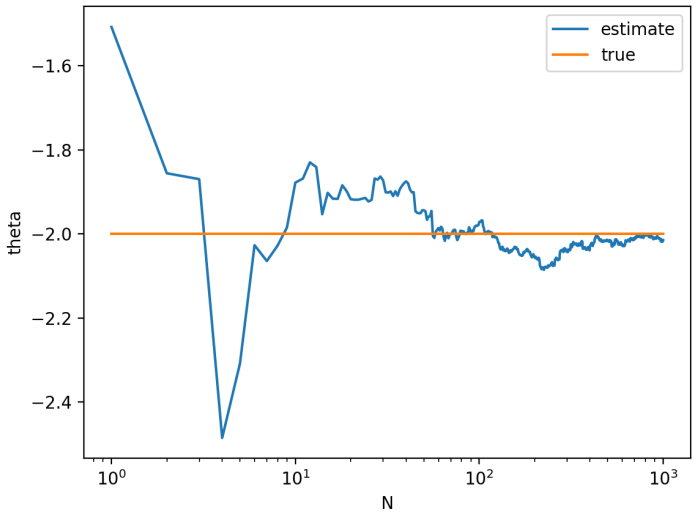

図 16: $\hat{\theta}_N$ の第二成分の挙動. 青線が推定値で橙線が真値. 縦軸が $\hat{\theta}_N$ の第二成分、横軸が用いたデータ数.(ただし log スケールである)

26　この課題のまとめと考察

式 (19.3) を用いた場合の方が推定されたパラメータが真の値 (3,-2) に近かった。また、プロットから分かる通り、式 (19.3) を用いた場合はパラメータを収束させることに成功している一方、式 (0.3) を用いた場合は収束しているとは言い難い結果となった。これらは課題 12 で述べたように、重み付き最小二乗法の優位性を示すものであると言える。
また、課題 12 の場合とは異なり、式 (19.3) を用いた場合であっても、与えられたデータの多くを使わなければパラメータは収束しなかった。課題 12 との相違点は、$i = 501, ..., 1000$ のデータにおける観測誤差の従う分布の分散が V_2 であることのみであるため、これが原因であると推察できる。

27　コード

コード 6: 課題 13 のコード

```
1 import pandas as pd
2 import numpy as np
3 import numpy.linalg as la
4 import matplotlib.pyplot as plt
```

```
5
6  n = 2
7  N = 1000
8  V_1 = np.array([[100.0, 0.0],[0.0, 1.0]])
9  V_2 = np.array([[2.0, 0.0],[0.0, 1.0]])
10
11 #基底関数
12 def phi(x):
13     kitei = np.array([[x**0, x**0],[x, x**2]])
14     return kitei.T
15
16 #6_5 の方法でパラメータを計算し、推定誤差分散を求める関数
17 def calc_theta_6_5(x, y,data_num):
18     Phi = np.array([[0.0,0.0],[0.0,0.0]])
19     for i in range(data_num):
20         Phi += np.dot(phi(x)[i].T, phi(x)[i])
21     Phi = la.inv(Phi)
22
23     migi = np.array([0.0,0.0])
24     for i in range(data_num):
25         migi += np.dot(phi(x)[i].T, y[i])
26
27     theta = np.dot(Phi, migi)
28
29     Err_mat = np.array([[0.0,0.0],[0.0,0.0]])
30
31     for i in range(data_num):
32         #V により場合分け
33         if i < 500:
34             phi_v = np.dot(phi(x)[i].T, V_1)
35         else:
36             phi_v = np.dot(phi(x)[i].T, V_2)
37         phi_v_phi = np.dot(phi_v, phi(x)[i])
38         phi_v_phi_Phi = np.dot(phi_v_phi, Phi)
39         Err_mat += phi_v_phi_Phi
40     Err_mat = np.dot(Phi, Err_mat)
41
42     return theta, Err_mat
43
44 #6_17 によりパラメータと推定誤差分散を計算する関数
45 def calc_theta_6_17(x, y, data_num):
46     Phi = np.array([[0.0,0.0],[0.0,0.0]])
47     for i in range(data_num):
48         if i < 500:
49             Q = la.inv(V_1)
50         else:
51             Q = la.inv(V_2)
52         phi_q = np.dot(phi(x)[i].T, Q)
53         Phi += np.dot(phi_q, phi(x)[i])
54     Phi = la.inv(Phi)
55
56     migi = np.array([0.0,0.0])
57     for i in range(data_num):
58         if i < 500:
59             Q = la.inv(V_1)
60         else:
61             Q = la.inv(V_2)
62         phi_q = np.dot(phi(x)[i].T, Q)
63         migi += np.dot(phi_q, y[i])
64
65     theta = np.dot(Phi, migi)
66
67     #推定誤差分散を計算
```

```
68      Err_mat = np.array([[0.0,0.0],[0.0,0.0]])
69
70      for i in range(data_num):
71          #V により場合わけ
72          if i < 500:
73              phi_v = np.dot(phi(x)[i].T, V_1)
74          else:
75              phi_v = np.dot(phi(x)[i].T, V_2)
76          phi_v_phi = np.dot(phi_v, phi(x)[i])
77          phi_v_phi_Phi = np.dot(phi_v_phi, Phi)
78          Err_mat += phi_v_phi_Phi
79      Err_mat = np.dot(Phi, Err_mat)
80
81      return theta, Err_mat
82
83  #データ読み込み
84  df = pd.read_csv("./suri_jikken6_data/mmse_kadai6.txt",header=
        None)
85  data = np.array(df)
86
87  x = np.array(data[:,0])
88  y = np.array(data[:,1:3])
89
90  ans_6_5 = calc_theta_6_5(x, y, N)
91  print(ans_6_5[0])
92
93  ans_6_17 = calc_theta_6_17(x, y, N)
94  print(ans_6_17[0])
95
96  #プロットのためのリスト
97  theta_0_list = []
98  theta_1_list = []
99  theta_2_list = []
100 theta_3_list = []
101 N_list = []
102 real_ans_0_list = []
103 real_ans_1_list = []
104 real_ans_2_list = []
105 real_ans_3_list = []
106
107 #データ数を変えてプロット
108 for data_num in range(1, N+1):
109     N_list.append(data_num)
110     real_ans_0_list.append(3.0)
111     real_ans_1_list.append(−2.0)
112     real_ans_2_list.append(3.0)
113     real_ans_3_list.append(−2.0)
114
115     x = np.array(data[0:data_num,0])
116     y = np.array(data[0:data_num,1:3])
117     #print(x.shape)
118
119     theta_0 = calc_theta_6_5(x, y, data_num)[0][0]
120     theta_1 = calc_theta_6_5(x, y, data_num)[0][1]
121     theta_2 = calc_theta_6_17(x, y, data_num)[0][0]
122     theta_3 = calc_theta_6_17(x, y, data_num)[0][1]
123
124     theta_0_list.append(theta_0)
125     theta_1_list.append(theta_1)
126     theta_2_list.append(theta_2)
127     theta_3_list.append(theta_3)
128
129 fig, ax = plt.subplots(facecolor="w")
```

```
ax.plot(N_list, theta_0_list, label="estimate")
ax.plot(N_list, real_ans_0_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_1_list, label="estimate")
ax.plot(N_list, real_ans_1_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_2_list, label="estimate")
ax.plot(N_list, real_ans_2_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()

fig, ax = plt.subplots(facecolor="w")
ax.plot(N_list, theta_3_list, label="estimate")
ax.plot(N_list, real_ans_3_list, label="true")
ax.legend()

plt.xscale('log')
plt.xlabel('N')
plt.ylabel('theta')
plt.show()
```

第 X 部

課題 14

$x_i \in \mathbb{R}, i = 1, ..., 10000$ に対し、次の観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = \varphi(x_i)\theta + w_i \tag{27.1}$$

ここで、

$$\varphi(x) = [1 \quad exp(-\frac{(x_i - 1)^2}{2}) \quad exp(-(x_i + 1)^2)] \tag{27.2}$$

で、w_i は平均 0、分散有限の独立同一分布から生成されているとする。この時、10000 組のデータ $\{(x_i, y_i)\}, i = 1, ..., 10000$ のうち、最初の 6000 組と残りの 4000 組のデータそれぞれで θ の推定値を求め、それらから推定値を合

成した。また、全データを使用した場合の推定値と比較し、一致することを確かめた。

28　準備

この課題を解くのに必要な前提知識について述べる。

28.1　異なるデータセットからの推定値の合成

二組の異なるデータセットからそれぞれ導き出されたパラメータの推定値を合成し、パラメータを更新することができる。データ数が M、N の 2 つのデータセットに対し、最小二乗法を用いて

$$\begin{aligned}\hat{\theta}_N &= (\sum_{i=1}^{N}\varphi(x_i)^T\varphi(x_i))^{-1}\sum_{j=1}^{N}\varphi(x_j)^T y_j \\ \hat{\theta}_M &= (\sum_{i=1}^{M}\varphi(x_i)^T\varphi(x_i))^{-1}\sum_{j=1}^{M}\varphi(x_j)^T y_j\end{aligned} \tag{28.1}$$

のように推定値を得ることができる。これを合成すると、

$$\begin{aligned}&\hat{\theta}_{N+M} \\ &= (\Phi_N^{-1}+\Phi_M^{-1})^{-1}\sum_{i=1}^{N+M}\varphi_i^T y_i \\ &= (\Phi_N^{-1}+\Phi_M^{-1})^{-1}\sum_{i=1}^{N}\varphi_i^T y_i + \\ &\quad (\Phi_N^{-1}+\Phi_M^{-1})^{-1}\sum_{i=N+1}^{N+M}\varphi_i^T y_i \\ &= (\Phi_N^{-1}+\Phi_M^{-1})^{-1}\Phi_N^{-1}\Phi_N\sum_{i=1}^{N}\varphi_i^T y_i + \\ &\quad (\Phi_N^{-1}+\Phi_M^{-1})^{-1}\Phi_M^{-1}\Phi_M\sum_{i=N+1}^{N+M}\varphi_i^T y_i \\ &= (\Phi_N^{-1}+\Phi_M^{-1})^{-1}(\Phi_N^{-1}\hat{\theta}_N+\Phi_M^{-1}\hat{\theta}_M)\end{aligned} \tag{28.2}$$

となる。これより、Φ_N と $\hat{\theta}$ を保存することで後から推定値の合成を行えるようになることが分かる。

29　課題 14 の回答

最初の 6000 組のデータで求めた推定値、残りの 4000 組のデータで求めた推定値、両者を合成した推定値、全データを用いた場合の推定値はそれぞれ以下のようになった。表より、確かに合成して求めた推定値と全てのデータ

表 11: 上から最初の 6000 組のデータで求めた推定値、残りの 4000 組のデータで求めた推定値、両者を合成した推定値、全データを用いた場合の推定値となっている.

最初の 6000 組のデータで求めた推定値	(-0.00387938 3.0107149 -1.98943435)
残りの 4000 組のデータで求めた推定値	(-0.02537589 3.03489937 -1.97772731)
合成して求めた推定値	(-0.0124955 3.02042997 -1.98486916)
全てのデータで求めた推定値	(-0.0124955 3.02042997 -1.98486916)

を用いて求めた推定値は一致することが確認できた。

30　この課題のまとめと考察

以上の結果から、合成して求めた推定値と全てのデータを用いて求めた推定値は一致することが確認できた。

31　コード

コード 7: 課題 14 のコード

```
1  import pandas as pd
2  import numpy as np
3  import numpy.linalg as la
4  import matplotlib.pyplot as plt
5
6  def phi(x):
7      kitei = np.array([x**0, np.exp(-(x-1.0)**2/2.0), np.exp(-(x+1.0)
           **2)])
8      return np.transpose(kitei)
9
10 def calc_theta(x, y):
11     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
12     ans_kari = np.dot(Phi, np.transpose(phi(x)))
13     ans = np.dot(ans_kari, y)
14     return ans
15
16 df = pd.read_csv("./suri_jikken6_data/mmse_kadai7.txt",header=
       None)
17 data = np.array(df)
18
19 N = 6000
20 M = 4000
```

```
#始めの組 6000
x = np.array(data[0:N,0])
y = np.array(data[0:N,1])
Phi_N = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
#print(Phi_N.shape)

N_ans = calc_theta(x, y)
print(N_ans)

#後の組 4000
x = np.array(data[N:,0])
y = np.array(data[N:,1])
Phi_M = la.inv(np.dot(np.transpose(phi(x)),phi(x)))

M_ans = calc_theta(x, y)
print(M_ans)

#合成
invN = la.inv(Phi_N)
invM = la.inv(Phi_M)
gousei_hidari = la.inv(invM + invN)
gousei_migi = np.dot(invN, N_ans) + np.dot(invM, M_ans)
gousei_ans = np.dot(gousei_migi, gousei_hidari)
print(gousei_ans)

x = np.array(data[:,0])
y = np.array(data[:,1])
ans = calc_theta(x, y)
print(ans)
```

第 XI 部

課題 15

$x_i \in \mathbb{R}, i = 1, ..., 10000$ に対し、次の観測データ $y_i \in \mathbb{R}$ が生成されているとする。

$$y_i = \varphi(x_i)\theta + w_i \tag{31.1}$$

ここで、

$$\varphi(x) = [1 \quad exp(-\frac{(x_i - 1)^2}{2}) \quad exp(-(x_i + 1)^2)] \tag{31.2}$$

であり、w_i は最初の 6000 組と 4000 組のデータで、平均 0 の異なる独立な分布に従うものとする。この時、最初の 6000 組と残りの 4000 組のデータそれぞれで θ の推定値と偶然誤差 w_i の分散の推定値を求め、推定された分散を用いて推定値を合成した。また、全データを用いて式 (0.3) により得た $\hat{\theta}_{10000}$ とどちらが優れているか比較、考察した。

32　準備

この課題を解くのに必要な前提知識について述べる。

32.1　重み行列を用いた推定値の合成

観測誤差の大きさが異なる場合、重み付きの最小二乗法を用いた方が結果の精度が上がる。データ数が N_1, N_2 のデータセットが二組与えられている時、重み行列 Q_1, Q_2 を使ったときのそれぞれのデータセットにおける推定値を $\hat{\theta}_{N_1}, \hat{\theta}_{N_2}$ とおくと、

$$\hat{\theta}_{N_1+N_2} = (\Phi^{-1}_{Q_1,N_1} + \Phi^{-1}_{Q_2,N_2})^{-1} \times (\Phi^{-1}_{Q_1,N_1}\hat{\theta}_{N_1} + \Phi^{-1}_{Q_2,N_2}\hat{\theta}_{N_2}) \qquad (32.1)$$

となる。ただし

$$\Phi_{Q,N} = (\sum_{i=1}^{N} \varphi_i^T Q \varphi_i)^{-1}$$

である。なお、式 (32.1) が成立することを示したのが問題 2 である。

33　課題 15 の回答

最初の 6000 組から求めたパラメータの推定値と偶然誤差 w_i の分散の推定値、残りの 4000 組から求めたパラメータの推定値と偶然誤差 w_i の分散の推定値、推定された分散を用いて合成したパラメータの推定値は以下のようになった。

表 12: 上から順に最初の 6000 組から求めたパラメータの推定値、偶然誤差 w_i の分散の推定値

パラメータの推定量
(0.00707684 3.28054335 -2.1908997)
偶然誤差の分散の推定値
96.86329354733162

表 13: 上から順に残りの 4000 組から求めたパラメータの推定値、偶然誤差 w_i の分散の推定値

パラメータの推定量
(0.09848311 3.10040485 -2.09100212)
偶然誤差の分散の推定値
0.010304240740875456

表 14: 上から順に推定された分散から合成した推定値、全データを用いて求めた推定値

合成した推定値
(0.09846859 3.10043371 -2.09101821)
全データを用いて求めた推定値
(0.04373209 3.20866562 -2.15117856)

34 この課題のまとめと考察

パラメータの真値は (0.1,3.1,-2.1) である。全データを用いて求めた推定値より前半と後半にデータを分けてから合成した推定値の方が、全ての要素において真値に近いことから、後者の方が優れた推定であるということがいえる。

35 コード

コード 8: 課題 15 のコード

```
1  import pandas as pd
2  import numpy as np
3  import numpy.linalg as la
4  import matplotlib.pyplot as plt
5
6  def phi(x):
7      kitei = np.array([x**0, np.exp(-(x-1.0)**2/2.0), np.exp(-(x+1.0)
           **2)])
8      return np.transpose(kitei)
9
10 def calc_theta(x, y):
11     Phi = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
12     ans_kari = np.dot(Phi, np.transpose(phi(x)))
13     ans = np.dot(ans_kari, y)
14     return ans
15
16 df = pd.read_csv("./suri_jikken6_data/mmse_kadai8.txt",header=
       None)
17 data = np.array(df)
18
19 N = 6000
20 M = 4000
21 n = 3
22 #始めの組 6000
23 x = np.array(data[0:N,0])
24 y = np.array(data[0:N,1])
25 Phi_N = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
26
27 N_ans = calc_theta(x, y)
28 print("前半の推定値:",N_ans)
29 V_n_hat_kari = y - np.dot(phi(x), N_ans)
30 V_n_hat_kari = np.dot(V_n_hat_kari, np.transpose(V_n_hat_kari))
31 V_n_hat_N = V_n_hat_kari / (N - n)
32 print("前半の推定誤差:",V_n_hat_N)
```

```
33
34 kari = np.dot(phi(x).T, V_n_hat_N)
35 Phi_Q_N = la.inv(np.dot(kari, phi(x)))
36
37 #後の組 4000
38 x = np.array(data[N:,0])
39 y = np.array(data[N:,1])
40 Phi_M = la.inv(np.dot(np.transpose(phi(x)),phi(x)))
41
42 M_ans = calc_theta(x, y)
43 print("後半の推定値:",M_ans)
44 V_n_hat_kari = y − np.dot(phi(x), M_ans)
45 V_n_hat_kari = np.dot(V_n_hat_kari, np.transpose(V_n_hat_kari))
46 V_n_hat_M = V_n_hat_kari / (M − n)
47 print("後半の推定誤差:",V_n_hat_M)
48
49 kari = np.dot(phi(x).T, V_n_hat_M)
50 Phi_Q_M = la.inv(np.dot(kari, phi(x)))
51
52 #合成
53 Phi1 = Phi_N * V_n_hat_N
54 Phi2 = Phi_M * V_n_hat_M
55 gousei_ans = la.inv(la.inv(Phi1)+ la.inv(Phi2)).dot(la.inv(Phi1).dot(N_ans
       )+la.inv(Phi2).dot(M_ans))
56 print("合成の推定値:",gousei_ans)
57 x = np.array(data[:,0])
58 y = np.array(data[:,1])
59 ans = calc_theta(x, y)
60 print("全データの推定値:",ans)
```

第 XII 部

課題 16

ここでは、次のバネ、マス、ダンパ系を表す直線状の運動方程式を考える。

$$M\frac{d^2}{dt^2}y(t) + D\frac{d}{dt}y(t) + Ky(t) = F(t) \tag{35.1}$$

ここで、M, K, D は正の定数で、$F(t)$ は外力である。このとき、与えられたデータを用い、様々な外力の場合について (M, K, D) を決定した。(M, K, D) の真の値は (2,1,3) とした。

36　準備

この課題を解くのに必要な前提知識を述べる。

36.1 逐次最小二乗法

N 個のデータから推定された $\hat{\theta}_N$ と新たなデータ (x_{N+1}, y_{N+1}) を用いて、再帰的に $\hat{\theta}_{N+1}$ を求める。式 (0.3) より、次のように式変形が可能である。

$$\begin{aligned}
&\hat{\theta}_{N+1} \\
&= \Phi_{N+1} \sum_{j=1}^{N+1} \varphi_j^T y_j \\
&= (\Phi_N^{-1} + \varphi_{N+1}^T \varphi_{N+1})^{-1} \times (\varphi_{N+1}^T y_{N+1} + \sum_{j=1}^{N} \varphi_j^T y_j) \\
&= \{\Phi_N - \Phi_N \varphi_{N+1}^T \times (I_m + \varphi_{N+1} \Phi_N \varphi_{N+1}^T)^{-1} \varphi_{N+1} \Phi_N\} \times (\varphi_{N+1}^T y_{N+1} + \sum_{j=1}^{N} \varphi_j^T y_j) \\
&= \hat{\theta}_N + K_{N+1}(y_{N+1} - \varphi_{N+1} \hat{\theta}_N)
\end{aligned} \tag{36.1}$$

ただし

$$K_{N+1} = \Phi_N \varphi_{N+1}^T (I_m + \varphi_{N+1} \Phi_N \varphi_{N+1}^T)^{-1}$$

となる。以上で得られた式および $\tilde{\Phi}_0 = \frac{1}{\epsilon} I_n$($\epsilon$ は小さな正の定数) から、以下の更新式を得られる。

$$\begin{aligned}
\tilde{\theta}_{N+1} &= \tilde{\theta}_N + \tilde{K}_{N+1}(y_{N+1} - \varphi_{N+1} \tilde{\theta}_N) \\
\tilde{K}_{N+1} &= \tilde{\Phi}_{N+1}^T (I_m + \varphi_{N+1} \tilde{\Phi_N} \varphi_{N+1}^T)^{-1} \\
\Phi_{N+1} &= \tilde{\Phi}_N - \tilde{K}_{N+1} \varphi_{N+1} \tilde{\Phi}_N
\end{aligned} \tag{36.2}$$

ただし

$$\tilde{\theta}_0 = 0, \tilde{\Phi}_0 = \frac{1}{\epsilon} I_n$$

とした。

37 課題 16 の回答

式 (35.1) で表される運動方程式を考える。この際、微小な δt では

$$\begin{aligned}
\frac{d}{dt} y(t) &= \frac{y((k+1)\delta t) - y(k\delta t)}{\delta t} \\
\frac{d^2}{dt^2} y(t) &= \frac{y((k+2)\delta t) - 2y((k+1)\delta t) + y(k\delta t)}{\delta t^2}
\end{aligned} \tag{37.1}$$

と近似できることを利用し、$y_k = y(k\delta t), F_k = F(k\delta t)$ とすると、

$$\begin{aligned} & y_k \\ &= (2 - \frac{D}{M}\delta t)y_{k-1} \\ &+(-1 + \frac{D}{M}\delta t - \frac{K}{M}\delta t^2)y_{k-2} \\ &+\frac{\delta t^2}{M}F_{k-2} + w_k \end{aligned} \tag{37.2}$$

を得る (w_k は近似により生まれる誤差)。以上より、問題は

$$y_k = x_k^T\theta + w_k, x_k = \begin{pmatrix} y_{k-1} \\ y_{k-2} \\ F_{k-2} \end{pmatrix} \tag{37.3}$$

の θ を求める問題に帰着する。本問では、$\delta t = 0.01$ とし、w_k は [-1,1] の一様分布に従う各時刻で独立な確率変数であるとした。

37.1 1.

ここでは $y_{-2} = 0, y_{-1} = 0$ とし、$F_k = 1, k = -2, -1, 0, 1, 2, ...$ として一定の外力を加えたとき、パラメータ θ を逐次最小二乗法で求めた。パラメータの初期値は 0 とし、$k = 10000$ まで計算した。また、正則化項において $\epsilon = 10^{-6}$ とした。結果、もとまったパラメータは以下のようになった。

表 15: 課題 16.1 で推定されたパラメータ

パラメータの推定量
(1.99436132 -0.99450446 -0.00491503)

37.2 2.

ここでは $y_{-2} = 0, y_{-1} = 0$ とし、$F_k = sin(\pi k/5), k = -2, -1, 0, 1, 2, ...$ として一定の外力を加えたとき、パラメータ θ を逐次最小二乗法で求めた。パラメータの初期値は 0 とし、$k = 10000$ まで計算した。また、正則化項において $\epsilon = 10^{-6}$ とした。

表 16: 課題 16.2 で推定されたパラメータ

パラメータの推定量
(1.99568564 -0.99585218 0.02102184)

37.3 3.

ここでは $y_0 = 0, y_1 = 0$ とし、どのような外力を加えればパラメータの推定がうまくいくか考え実装し、逐次最小二乗法で求めた。パラメータの初期値は 0 とし、$k = 10000$ まで計算した。
具体的には、外力を 10^8 で一定にしたものと、$k = 0$ の時のみ 10^8 の力をかけ、あとは全て 0 で一定にしたものを実装した。以下が結果の表である。

表 17: 課題 16.3 で推定されたパラメータ. 上から順に 10^8 の外力をかけ続けたもの、はじめのみ 10^8 の外力をかけて後は 0 としたもの

10^8 の外力をかけ続けた場合
(1.99500051e+00 -9.95150492e-01 4.99939377e-05)
はじめのみ 10^8 の外力をかけて後は 0 とした場合
(1.99498920e+00 -9.95139267e-01 4.99953467e-05)

38 この課題のまとめと考察

パラメータの真の値は式 (37.2) に K,M,D,δt を代入することで求められる。y_{k-1} の係数が $2 - \frac{D}{M}\delta t$ で、これがパラメータの第一成分にあたる。同様に第二成分が $-1 + \frac{D}{M}\delta t - \frac{K}{M}\delta t^2$、第三成分が $\frac{\delta t^2}{M}$ となるため、これに K,M,D,δt を代入して、真の値は (1.995,-0.99515,0.00005) となる。この真値と比較することにより、1,2 で推定したパラメータはあまり精度が高くないことがわかった。一方、3 で推定したパラメータは真の値に近く、正しい推定ができていると結論づけることができた。
正則化項における ϵ は、10^{-4} から 10^{-10} 程度まで値を変えて調べたものの、問 1,2 共に結果にはほぼ変化がなかった。
3 で正しい推定ができる理由は、加えた外力が近似の際に生じる誤差と比較して十分大きいからであると考えられる。
ただし、今回採用した外力はどちらも非現実的である。前者は外力が強すぎて系を破壊しかねず、後者は 0 秒で大きな力を加える、すなわちインパルス応答となっているためである。ただ、後者に関しては衝撃波を加えることにより、近い状態を作り出すことは可能かもしれない。

39 コード

16.1、16.2、16.3 のコードの相違点は F の定義のみであるため、ここでは 1 のコードのみ掲載する。

コード 9: 課題 16.1、2、3 のコード

```
from numpy.core.fromnumeric import reshape
import pandas as pd
import numpy as np
import numpy.linalg as la

#使ってない
def phi(x):
    return x

f = open("16_1_y.txt", "w")
m = 3

#逐次最小二乗法
def saisyou_nijou(theta, Phi, x, y):
    x = x.reshape(1,3)
    K = Phi.dot(x.T).dot(la.inv(1.0+x.dot(Phi.dot(x.T))))
    #print(K.shape)
    theta = theta + K.dot(y - x.dot(theta))
    Phi = Phi - K.dot(x).dot(Phi)
    return theta, Phi

#外力
def F(k):
    return 1.0#2 番では np.sin(np.pi*k/5.0)に変える

#データ生成
def y_k_real(y_pre, y_pre_pre, k):
    M, D, K, dt = 2.0, 1.0, 3.0, 0.01
    w = np.random.rand()*2.0-1.0
    #w = 0.0
    #print(w)
    y_new = (2.0 - (D/M)*dt)*y_pre + (-1.0 + D/M*dt - (K/M)*(dt
        **2)) * y_pre_pre + ((dt**2)/M) * F(k-2) + w
    return y_new

#パラメータ
epsilon = 10 **(-6)
y_pre = 0.0
y_pre_pre = 0.0
theta = np.array([0.0,0.0,0.0])
I = np.eye(m)
Phi = I/epsilon

#実行
for k in range(10001):
    y = y_k_real(y_pre, y_pre_pre, k)
    f.writelines([str(y),"\n"])
    x = np.array([y_pre, y_pre_pre, F(k-2)])
    theta, Phi = saisyou_nijou(theta, Phi, x, y)
    y_pre_pre = y_pre
    y_pre = y

f.close()
print(theta)
```

第 XIII 部
課題 17

次の正弦波で変動する信号を考える。

$$y_k = sin(0.0001k) + w_k, k = 1, ..., 10000 \tag{39.1}$$

ただし w_k は+1 と-1 を確率 $\frac{1}{2}$ で出すベルヌーイ過程であるとし、$\theta_k = sin(0.0001k)$ は未知であるとする。このとき、$\gamma = 0.99$ の忘却係数付きの逐次最小二乗法で θ_k を推定し、各時刻の $\hat{\theta_{\gamma,k}}$ をプロットした。

40　準備

この課題を解くために必要な前提知識を述べる。

40.1　重みつき逐次最小二乗法

重み行列が単位行列の定数倍 ($Q_i = \rho_i I$) であるとき、更新式は以下のようになる。

$$\begin{aligned} \hat{\theta}_{N+1} &= \hat{\theta}_N + K_{N+1}(y_{N+1} - \varphi_{N+1}\hat{\theta}_N) \\ K_{N+1} = \rho_{N+1}\Phi_N\varphi_{N+1}^T &\times (I_m + \rho_{N+1}\varphi_{N+1}\Phi_N\varphi_{N+1}^T)^{-1} \\ \Phi_{N+1} &= \Phi_N - K_{N+1}\varphi_{N+1}\Phi_N \end{aligned} \tag{40.1}$$

時系列データの場合、新しいデータの影響を大きく使いたいことが多い。そこで、N 組のデータがあるとき、$\rho_i = \gamma^{N-i}, \gamma \in (0, 1)$ とおくと、その推定値は

$$\hat{\theta}_{\gamma,N} = (\sum_{i=1}^{N} \gamma^{N-i}\varphi_i^T\varphi_i)^{-1} \sum_{j=1}^{N} \gamma^{N-j}\varphi_j^T y_j \tag{40.2}$$

となる。同様に $N+1$ 組のデータでは

$$\hat{\theta}_{\gamma,N+1} = (\varphi_{N+1}^T\varphi_{N+1} + \gamma\sum_{i=1}^{N} \gamma^{N-i}\varphi_i^T\varphi_i)^{-1} \times (\varphi_{N+1}^T y_{N+1} + \gamma\sum_{j=1}^{N} \gamma^{N-j}\varphi_j^T y_j) \tag{40.3}$$

となるため、これを整理することで、次の更新式を得る。

$$\begin{aligned} \hat{\theta}_{\gamma,N+1} &= \hat{\gamma,}\theta_N + K_{\gamma,N+1}(y_{N+1} - \varphi_{N+1}\hat{\gamma,}\theta_N) \\ K_{\gamma,N+1} = \rho_{N+1}\Phi_N\varphi_{N+1}^T &\times (I_m + \rho_{N+1}\varphi_{N+1}\Phi_{\gamma,N}\varphi_{N+1}^T)^{-1} \\ \Phi_{\gamma,N+1} &= \frac{1}{\gamma}\Phi_{\gamma,N} - \frac{1}{\gamma}K_{\gamma,N+1}\varphi_{N+1}\Phi_{\gamma,N} \end{aligned} \tag{40.4}$$

この γ を忘却係数という。

41　課題 17 の回答

プロットは以下のようになった。横軸は時刻、縦軸は $\hat{\theta}_{\gamma,k}$ である。

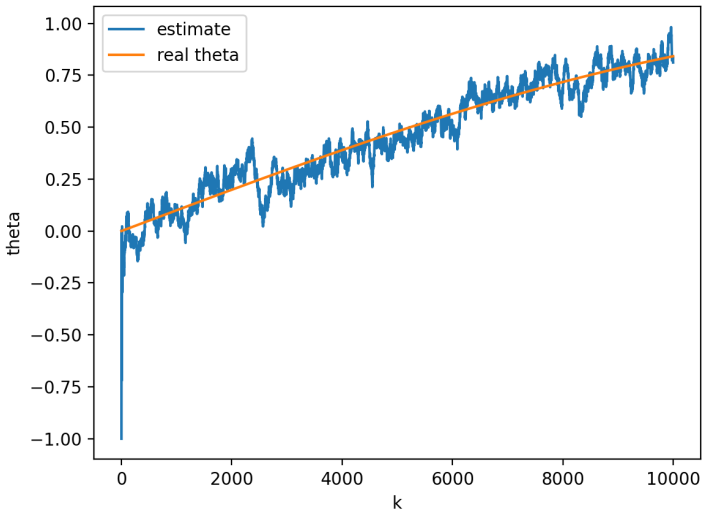

図 17: 各時刻における $\hat{\theta}_{\gamma,k}$ の様子. 横軸が時刻、縦軸が $\hat{\theta}_{\gamma,k}$ である. 青線が予測値、橙線が真の値.

42　この課題のまとめと考察

プロットより、実際の $sin(0.0001k), k = 1, ..., 10000$ と予測値が概ね一致していることが分かるので、この予測の精度は高いと結論づけることができる。

43　コード

コード 10: 課題 17 のコード

```
1
2  from numpy.random import random
3  import pandas as pd
4  import numpy as np
5  import numpy.linalg as la
6  import matplotlib.pyplot as plt
7
8  #忘却係数
9  gamma = 0.99
10
11 #実際の y を生成する関数
```

```
def y_k_real(k):
    belnui = np.random.randint(0,2)
    if belnui == 0:
        w = −1.0
    else:
        w = 1.0
    return np.sin(0.0001 ∗ k) + w

#忘却最小二乗法の関数
def boukyaku_saisyou_nijou(Phi, theta, y):
    K = Phi / (1.0 + Phi)
    theta = theta + K ∗ (y − theta)
    Phi = Phi / gamma − K ∗ Phi / gamma
    return theta, Phi

#パラメータ
theta = 0.0
epsilon = 10 ∗∗ (−6)
Phi = 1.0 / epsilon
theta_list = []
k_list = []
sink_list = []

#それぞれの k に対して予測値を計算
for k in range(1, 10001):
    k_list.append(k)
    y = y_k_real(k)
    theta, Phi = boukyaku_saisyou_nijou(Phi, theta, y)
    theta_list.append(theta)
    sink_list.append(np.sin(k ∗ 0.0001))

#プロット
fig, ax = plt.subplots(facecolor="w")
ax.plot(k_list, theta_list, label="estimate")
ax.plot(k_list, sink_list, label="real␣theta")
ax.legend()

plt.xlabel("k")
plt.ylabel("theta")
plt.show()
```

第 XIV 部

課題 18

次の観測方程式が与えられているとする。

$$\begin{aligned} \theta_k &= 0.9\theta_{k-1} + v_k \\ y_k &= 2\theta_k + w_k \end{aligned} \tag{43.1}$$

ここで、$\theta_k, y_k \in \mathbb{R}$ であり、v_k, w_k は互いに独立な $\mathcal{N}(0,1)$ に従う確率変数である。また、θ_0 は平均 3、分散 2 を持つ確率分布に従うとする。このとき、$y_l, l = 1, ..., k$ までを使った $\theta_k, k = 1, ..., 100$ の推定値 $\hat{\theta_k}$ を求め、θ_k とともに時間変化をプロットした。

44　準備

この課題を回答するために必要な前提知識を述べる。

44.1　Kalman フィルタ

次の線形差分方程式を考える。

$$\begin{aligned} \theta_k &= a_k\theta_{k-1} + v_k \\ y_k &= c_k\theta_k + w_k, k = 1, 2, ... \end{aligned} \tag{44.1}$$

ここで、$a_k, c_k \in \mathbb{R}$ であり、$v_k, w_k \in \mathbb{R}$ は互いに独立な平均 0、分散がそれぞれ σ_v^2, σ_w^2 の Gauss 分布にしたがって発生し、さらに異なる時刻で独立であるとする。また、$c_k \neq 0, k = 1, ...,$ とする。θ_0 は平均 $\bar{\theta}$、分散 P を持つ分布に従って発生するとする。時刻 k までのデータによる推定値を $\hat{\theta}_k \in \mathbb{R}$ で表すとすると、式 (44.1) より

$$y_k = c_k(a_k\theta_{k-1} + v_k) + w_k \tag{44.2}$$

となるため、前の時刻の推定値を a_k 倍したものに新たなデータから修正する項を考え、次の規則で推定値を更新することを考える。

$$\hat{\theta}_k = a_k\hat{\theta}_{k-1}+F_k(y_k-c_ka_k\hat{\theta}_{k-1}) = a_k\hat{\theta}_{k-1}+a_kc_kF_k(\theta_{k-1}-\hat{\theta_{k-1}})+F_k(w_k+c_kv_k) \tag{44.3}$$

推定誤差分散を $V_k = \mathbb{E}[(\theta_k - \hat{\theta}_k)^2]$ とすると、

$$V_k = a_k^2(1 - c_kF_k)^2V_{k-1} + (1 - c_kF_k)^2\sigma_v^2 + \sigma_w^2F_k^2 \tag{44.4}$$

整理して

$$F_k = \frac{a_k^2c_kV_{k-1} + c_k\sigma_v^2}{a_k^2c_k^2V_{k-1} + c_k^2\sigma_v^2 + \sigma_w^2} \tag{44.5}$$

となる。変数 X_k を導入して見やすくすると、$\hat{\theta}_0 = \bar{\theta}, V_0 = P$ としてつぎの推定規則を導ける。

$$\begin{aligned} \hat{\theta}_k &= a_k\hat{\theta}_{k-1} + F_k(y_k - c_ka_k\hat{\theta}_{k-1}) \\ X_k &= a_k^2V_{k-1} + \sigma_v^2 \\ V_k = X_k - \frac{c_k^2X_k^2}{c_k^2X_k + \sigma_w^2} &= \frac{\sigma_w^2X_k}{c_k^2X_k + \sigma_w^2} \\ F_k &= \frac{c_kX_k}{c_k^2X_k + \sigma_w^2} \end{aligned} \tag{44.6}$$

以上で表される推定規則が Kalman フィルタである。

45　課題 18 の回答

ここでは、式 (44.6) において、$a_k = 0.9, c_k = 0.2, \sigma_w = \sigma_v = 1.0, \hat{\theta}_0 = 3, P = 2$ とすれば良い。また、θ_0 は平均 3、分散 2 の正規分布に従うものとした。結果のプロットは以下である。

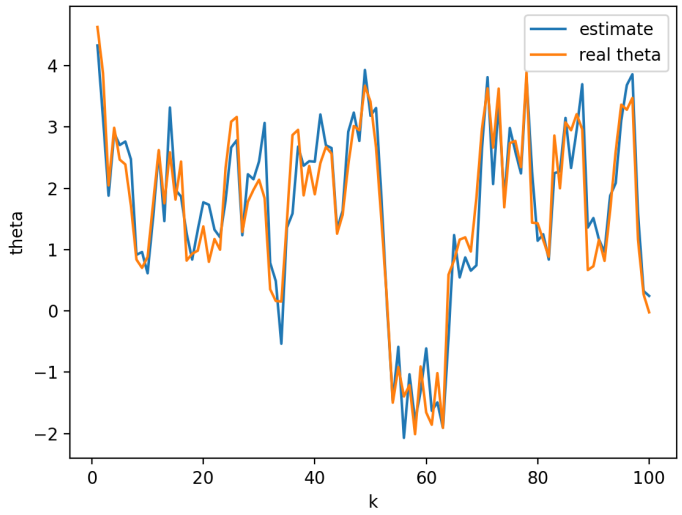

図 18: 課題 18 のプロット. 横軸時間、縦軸 $\hat{\theta}$ で、青線が予測値、橙線が実際のパラメータの値を表す.

46　この課題のまとめと考察

プロットより、パラメータの予測値および実際のパラメータの値の挙動がほぼ一致していることがわかる。この結果は、実装した Kalman フィルタが意図した通りの挙動をしていることを示すものである。

47　コード

コード 11: 課題 18 のコード

```
1 from numpy.random import random
2 import pandas as pd
3 import numpy as np
4 import numpy.linalg as la
5 import matplotlib.pyplot as plt
6
7 #パラメータ
```

```
a_k = 0.9
c_k = 2.0
theta = 3.0
theta_real = np.random.normal(loc=3.0, scale=2.0)
V_k = 2.0
theta_real_list = []
theta_list = []
k_list = []

#Kalman フィルタを繰り返し適用
for k in range(1,101):
    k_list.append(k)
    v_k = np.random.normal()
    theta_real = a_k * theta_real + v_k
    w_k = np.random.normal()
    y_k = c_k * theta_real + w_k
    X_k = (a_k ** 2) * V_k + 1.0
    V_k = 1.0 * X_k / ((c_k**2) * X_k + 1.0)
    F_k = c_k*X_k / ((c_k**2)*X_k + 1.0)
    theta = a_k * theta + F_k * (y_k - c_k * a_k * theta)
    theta_real_list.append(theta_real)
    theta_list.append(theta)

#プロット
fig, ax = plt.subplots(facecolor="w")
ax.plot(k_list, theta_list, label="estimate")
ax.plot(k_list, theta_real_list, label="real␣theta")
ax.legend()

plt.xlabel("k")
plt.ylabel("theta")
plt.show()
```

第 XV 部

課題 19

ここでは、課題 18 の設定の元で、初期値 θ_0 の推定値とその推定誤差分散を求めた。また、初期値の分散と比べて RTS アルゴリズムによる推定誤差分散がどれだけ改善できたかを推定誤差分散と初期分散の比で表した。

48　準備

この課題を解くのに必要な前提知識を述べる。

48.1　Kalman スムーザ

Kalman フィルタを用いた場合、y_k には過去の θ に関する情報も含まれている。これを利用し、新たなデータ y_k から推定値 $\hat{\theta}_l, l < k$ を更新するアルゴリズムが Kalman スムーザである。本問を解くのに用いたのはこのうち RTS

アルゴリズムと呼ばれるものである。
時刻 1 から N までのデータ $\{y_k\}$ から Kalman フィルタによる推定値 $\hat{\theta}_k$、推定誤差分散 V_k、一段先予測誤差分散 X_k(ただし $k = 1, ..., N$) が得られているとする。時刻 k の潜在変数 θ_k を全てのデータを用いて推定した結果を $\hat{\theta_k}^s$ で表す。この時、$\hat{\theta}^s_N = \hat{\theta}_N, V^s_N = V_N$ とした後退方程式

$$\begin{aligned} \hat{\theta}^s_k &= \hat{\theta}_k + g_k(\hat{\theta}^s_{k+1} - a_k\hat{\theta}_k) \\ g_k &= a_k \frac{V_k}{X_{k+1}} \\ V^s_k &= V_k + g^2_k(V_{k+1} - X_{k+1}) \end{aligned} \tag{48.1}$$

の解 $\hat{\theta}^s_k, k = 1, ..., N$ は時刻 N までの全ての情報を用いた最小平均二乗誤差推定値となる。また、V^s_k は推定誤差分散を表す。

49　課題 19 の回答

初期値 θ_0 の推定値、その推定誤差分散、推定誤差分散と初期分散の比はそれぞれ以下のようになった。

表 18: 上から順に初期値 θ_0 の推定値、その推定誤差分散、推定誤差分散と初期分散の比

初期値 θ_0 の推定値
5.048759074620913
推定誤差分散
0.8568473885739323
推定誤差分散と初期分散の比
0.42842369428696614

50　この課題のまとめと考察

推定誤差分散と初期分散の比がおよそ 0.43 となったことから、Kalman スムーザの適用により適切にパラメータの推定ができていることが確認できた。また、実際に θ_0 は平均 3、分散 2 の正規分布に従うものとしたことから、推定値も妥当であることがわかった。

51　コード

コード 12: 課題 19 のコード

```
from numpy.random import random
import pandas as pd
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt

#パラメータ
a_k = 0.9
c_k = 2.0
theta = 3.0
theta_real = np.random.normal(3.0, 2.0)
V_k = 2.0
theta_real_list = []
theta_list = [theta]
k_list = []
V_k_list = [V_k]
X_k_list = []

#Kalman フィルタを繰り返し適用
for k in range(1,101):
    k_list.append(k)
    v_k = np.random.normal()
    theta_real = a_k * theta_real + v_k
    w_k = np.random.normal()
    y_k = c_k * theta_real + w_k
    X_k = (a_k ** 2) * V_k + 1.0
    X_k_list.append(X_k)
    V_k = 1.0 * X_k / ((c_k**2) * X_k + 1.0)
    V_k_list.append(V_k)
    F_k = c_k*X_k / ((c_k**2)*X_k + 1.0)
    theta = a_k * theta + F_k * (y_k - c_k * a_k * theta)
    theta_real_list.append(theta_real)
    theta_list.append(theta)

theta_ks = theta
V_ks = V_k

#RSA アルゴリズムを適用
for i in reversed(range(100)):
    #print(i)
    g_k = a_k * V_k_list[i] / X_k_list[i]
    theta_ks = theta_list[i] + g_k * (theta_ks - a_k * theta_list[i])
    V_ks = V_k_list[i] + (g_k**2) * (V_ks - X_k_list[i])

#結果
print(theta_ks)
print(V_ks)
print(V_ks / 2.0)
```

第 XVI 部
課題 20

$x_i \in [-1,1], i = 1, ..., 10000$ に対し、次の方程式で定められる入出力データが与えられているとする。

$$y_i = \alpha^T \varphi(x_i)\beta + w_i \tag{51.1}$$

ここで、$\alpha \in \mathbb{R}^2, \beta \in \mathbb{R}^3, w_i$ は区間 [-1,1] の一様分布であるとし、

$$\varphi(x) = \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \end{pmatrix} \tag{51.2}$$

の場合を考える。この時、10000 組の (x_i, y_i) から交互最小二乗法を用いて α, β を推定した。また、10 組の異なる初期値を用意し、最もよかったパラメータとその時に使った初期値を示した。

52　準備

この課題を解くのに必要となる前提知識を述べる。

52.1　交互最小二乗法

関数 $f(\theta, x)$ が $\theta = (\alpha, \beta)^T$ に関して双線形である時、パラメータ θ を 2 つに分割すれば関数 f が α, β それぞれに対して線形になることから、$f(\theta, x)$ を改めて $g(\alpha, \beta, x)$ をおいて次のようにパラメータを求めることができる。これが交互最小二乗法である。

- 初期値 $\hat{\alpha}_0, \hat{\beta}_0$ を設定する。
- $j = 1, ...$ に対し、
  - $\beta = \hat{\beta}_{j-1}$ に固定し、α を次の式で求める。

$$\hat{\alpha}_j = \underset{\alpha \in \mathbb{R}^{n_1}}{arg\,min} \sum_{i=1}^{N} \|y_i - g(\alpha, \hat{\beta}_{j-1}, x_i)\|^2 \tag{52.1}$$

  - $\alpha = \hat{\alpha}_j$ に固定し、β を次の式で求める。

$$\hat{\beta}_j = \underset{\alpha \in \mathbb{R}^{n_2}}{arg\,min} \sum_{i=1}^{N} \|y_i - g(\hat{\alpha}_j, \beta, x_i)\|^2 \tag{52.1}$$

- 収束判定条件を満たすか判定し、満たさなければ j を一つ繰り上げて 2 つ目の手順にもどる。

各最小化問題は式 (0.3) で解けば良い。

53　課題 20 の回答

本問は交互最小二乗法を用いて回答した。交互最小二乗法の収束判定条件として、$\epsilon = 10^{-6}$ に対する

$$\|\alpha_{j-1} - \alpha_j\|^2 + \|\beta_{j-1} - \beta_j\|^2 < \epsilon$$

の成立を採用した。
交互最小二乗法における α, β の最小化の方法について述べる。
式 (0.1) で表される関数を最小化する θ は、式 (0.3) により求まる。本問で与えられている入出力データにおいては以下が成り立つ。

- β を固定した時
  - $\alpha^T \varphi(x_i)\beta = (\varphi(x_i)\beta)^T \alpha$ が成立する。
  - よって式 (0.3) において $\varphi(x_i)$ に $(\varphi(x_i)\beta)^T$ を代入して、

$$\{\sum_{i=1}^{N} (\varphi(x_i)\beta)(\varphi(x_i)\beta)^T\}^{-1} \sum_{j=1}^{N} (\varphi(x_j)\beta) y_j \tag{53.1}$$

    で求まる α が、与式を最小化する α である。

- α を固定した時
  - 式 (0.3) における $\varphi(x_i)$ に $\alpha^T \varphi(x_i)$ を代入して、

$$\{\sum_{i=1}^{N} (\alpha^T \varphi(x_i))^T \alpha^T \varphi(x_i)\}^{-1} \sum_{j=1}^{N} (\alpha^T \varphi(x_j))^T y_j \tag{53.2}$$

    で求まる β が、与式を最小化する β である。

10 組の初期値およびその時推定されたパラメータ、その中で最も良かったパラメータとその時に使った初期値は以下である。ただし、初期値はそれぞれ以下の分布からランダムに生成した。これらは、パラメータの真値が期待値である幅 2.0 の分布となるように設定した。

- α の第一成分：[0,2] の一様分布
- α の第二成分：[-3,-1] の一様分布
- β の第一成分：[-0.5,1.5] の一様分布
- β の第一成分：[-2.0,0] の一様分布
- β の第一成分：[1,3] の一様分布

また、最も良いパラメータとして、回帰モデルとデータの二乗誤差が最も小さくなるものを採用した。

表 19: 最も良かったパラメータとその時に使った初期値. 上から順にパラメータ、初期値である.

パラメータの推定値
α:(0.87907965 -1.74679254)β:(0.57773367 -1.13074314 2.29569288)
初期値
α:(0.88198351 -1.53665416)β:(0.28147475 -1.32918369 2.53393405)

表 20: 残り 9 組の初期値とその時のパラメータの推定値の表

初期値
α:(0.7676491 -2.54174922)β:(1.22429426 -0.65419038 2.28962945)
パラメータの推定値
α:(0.78894962 -1.55099889)β:(0.64660782 -1.25410935 2.59422949)
初期値
α:(0.62984476 -2.99070649)β:(1.42529549 -0.2928604 2.15821323)
パラメータの推定値
α:(0.81437652 -1.60245924)β:(0.62617943 -1.21545627 2.51018518)
初期値
α:(0.13134968 -2.26728033)β:(-0.37424283 -1.87392872 1.57321468)
パラメータの推定値
α:(1.2131248 -2.37282353)β:(0.42140601 -0.81369574 1.69847641)
初期値
α:(0.6294013 -2.312001)β:(1.2400193 -1.55985942 1.09836402)
パラメータの推定値
α:(0.79566288 -1.6260822)β:(0.63089796 -1.26200608 2.44527896)
初期値
α:(0.29370711 -2.84303114)β:(0.86280628 -1.045895 1.15930158)
パラメータの推定値
α:(1.03220684 -2.11844902)β:(0.48546665 -0.97405249 1.87463964)
初期値
α:(1.3178155 -2.58595513)β:(1.21250496 -1.94028981 1.88271072)
パラメータの推定値
α:(0.644844 -1.31297766)β:(0.77965378 -1.55533718 3.03170431)
初期値
α:(1.75905833 -1.60943668)β:(-0.30008309 -1.38187124 2.29619095)
パラメータの推定値
α:(1.10865399 -2.27165929)β:(0.45229525 -0.90644653 1.74902903)
初期値
α:(0.49026938 -1.70009918)β:(1.38500097 -0.11870411 1.51993166)
パラメータの推定値
α:(0.9850308 -1.93486129)β:(0.51807312 -1.00408518 2.08010615)
初期値
α:(0.99334458 -2.79502105)β:(0.67925781 -0.07801415 1.13280688)
パラメータの推定値
α:(1.69181861 -3.31293935)β:(0.30202616 -0.58377882 1.21604833)

54　この課題のまとめと考察

パラメータの真値が α:(1,-2)、β:(0.5,-1,2) であることから、9 組のパラメータはもちろん、最も良いパラメータに関してもあまり精度が高いとは言えなかった。交互最小二乗法を用いた場合、一般には局所最適解への収束は保証されない上、今回初期値はランダムに設定していることから、これは妥当な結果であると言える。

55　コード

コード 13: 課題 20 のコード

```
from os import spawnlp
from numpy.random import random
import pandas as pd
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt

#基底関数
def phi(x):
    return np.array([[x**0, x],[x, x**2],[x**2, x**3]]).T

epsilon = 10**(-4)
repeat_times = 10
df = pd.read_csv("./suri_jikken6_data/mmse_kadai13.txt",header=
    None)
data = np.array(df)
x = np.array(data[:,0])
y = np.array(data[:,1])
N = 10000
gosa_list = []

for time in range(repeat_times):
    #初期値をランダムに設定
    a_1 =np.random.rand()*2.0
    a_2 =np.random.rand()*2.0 - 3.0
    b_1 =np.random.rand()*2.0 - 0.5
    b_2 =np.random.rand()*2.0 - 2.0
    b_3 =np.random.rand()*2.0 + 1.0
    alpha = np.array([a_1, a_2])
    beta = np.array([b_1, b_2, b_3])
    alpha_syoki = alpha
    beta_syoki = beta
    alpha_pre = np.array([0.0, 0.0])
    beta_pre = np.array([0.0, 0.0, 0.0])
    j = 0

    #交互最小二乗法
    while np.sum((alpha_pre - alpha)**2) + np.sum((beta_pre - beta)
        **2) >= epsilon:
        alpha_pre = alpha
        beta_pre = beta
        j += 1
        hidari = np.array([[0.0, 0.0],[0.0, 0.0]])
        migi = np.array([0.0, 0.0])
        #α を求める
```

```
44          for i in range(N):
45              phi_beta = np.dot(phi(x)[i], beta)
46              migi += phi_beta * y[i]
47              phi_beta = np.reshape(phi_beta, (2,1))
48              hidari += np.dot(phi_beta, phi_beta.T)
49          hidari = la.inv(hidari)
50          alpha = np.dot(hidari, migi)
51
52          hidari = np.array([[0.0, 0.0, 0.0],[0.0, 0.0, 0.0],[0.0, 0.0, 0.0]])
53          migi = np.array([0.0, 0.0, 0.0])
54
55          #β を求める
56          for i in range(N):
57              alpha_reshape = np.reshape(alpha, (2, 1))
58              alpha_T_phi = np.dot(alpha_reshape.T, phi(x)[i])
59              alpha_T_phi_T = np.reshape(alpha_T_phi.T,(3,))
60              migi = migi + y[i] * (alpha_T_phi_T)
61              alpha_T_phi = np.reshape(alpha_T_phi, (1,3))
62              hidari += np.dot(alpha_T_phi.T, alpha_T_phi)
63          hidari = la.inv(hidari)
64          beta = np.dot(hidari, migi)
65
66      #誤差を計算
67      gosa = np.dot(np.dot(alpha.T, phi(x)), beta) − y
68      gosa = np.sum(gosa**2) / N
69      print("初期値
            ",alpha_syoki, beta_syoki,"alpha:",alpha,"beta:",beta,"誤差
            :",gosa)
70      gosa_list.append([gosa, alpha, beta, alpha_syoki, beta_syoki])
```

第 XVII 部

課題 21

本問では、k-means 法を用い、与えられたデータ $x_i \in \mathbb{R}^2, i = 1, ..., 1000$ を 3 つのクラスタに分類した。また、異なる初期値を 10 組選び、最も良さそうな分類をその理由とともに示した。

56　準備

この課題を解くのに必要な前提知識をまとめた。

56.1　K 平均法

$\{x_i\}, i = 1, ..., N \in \mathbb{R}^p$ がデータとして与えられている時、これらを K 個のクラスタに分類することを考える。$\nu = \{1, ..., N\}$ とし、ν の空でない部分集合 $\nu_1, ..., \nu_K$ を、$\cup_{l=1}^{K} \nu_l = \nu$ かつ $\nu_l \cap \nu_k = \phi, l \neq k$ を満たすようにとる。

この ν_l がクラスタを表し、そのクラスタを特徴づける量を次で定める。

$$\mu(\nu_l) = arg \min_{\mu \in \mathbb{R}^p} \sum_{i_l \in \mu_l} \|x_{i_l} - \mu\|^2, l = 1, ..., K \tag{56.1}$$

これは次のようにクラスタの中心を表す。

$$\mu(\nu_l) = \frac{1}{\|\nu_l\|} \sum_{i_l \in \nu_l} x_{i_l} \tag{56.2}$$

クラスタは、

$$r_{i,l} = \begin{cases} 1, i \in \nu_l \\ 0, i \notin \nu_l \end{cases} \tag{56.3}$$

を導入した上で、次の最小化問題を考えることで決定する。

$$\min_{\{\nu_1, ..., \nu_K\}} \sum_{l=1}^{K} \sum_{i=1}^{n} r_{i,l} \|x_{i_l} - \mu(\nu_l)\|^2 \tag{56.4}$$

x_i がどのクラスタに入るかを更新する際は、現在のクラスタ中心 $\mu(\nu_l)$ を固定して μ_l と表し、$r_{i,l}$ を更新する。この時、x_i は最も近い代表点を含むクラスタに所属するとして更新を行う。これは以下の式で表される。

$$r_{i,l} = \begin{cases} 1, l = arg \min_{k=1,...,K} \|x_i - \mu_k\|^2 \\ 0, otherwise \end{cases} \tag{56.5}$$

各クラスタに所属するデータが変更されたら、再度 $\mu(\nu_l)$ を計算し直す。以下が全体のアルゴリズムである。

- 初期値 $\mu_l \in \mathbb{R}^p, l = 1, ..., K$ を設定する。
- $j = 1, ...$ に対し,
  - 式 (56.5) より $r_{i,l}^{(j)}, i = 1, ..., N, l = 1, ..., K$ を定める。
  - 次の式より $\mu_l^{(j)}, l = 1, ..., K$ を定める。

$$\mu_l^{(j)} = \frac{1}{\sum_{i=1}^{N} r_{i,l}} \sum_{i=1}^{N} r_{i,l} x_i \tag{56.6}$$

- 収束判定条件を満たすか判定し、満たさなければ j を一つ繰り上げて二つ目の手順にもどる。

57 課題 21 の回答

本問では、収束判定条件として $\epsilon = 10^{-6}$ に対して $max_l \|\mu_l^{(j-1)} - \mu_l^{(j)}\|^2 < \epsilon$ の成立を採用した。また、10 通りの初期値 $\mu_l \in \mathbb{R}^2, l = 1, 2, 3$ の決め方は以下の通りとした。

- $\mu_l(l = 1, 2, 3)$ の第一成分は [-10,10] の一様分布からランダムに生成する

- $\mu_l(l = 1, 2, 3)$ の第二成分は [-5,5] の一様分布からランダムに生成する

これは与えられたデータをプロットして目視で確認し、3 つのクラスタに分割した場合に各クラスタの代表点となりそうな座標に目星をつけて定めた範囲である。
10 通りの分類結果は以下である。ただし、それぞれの場合の初期値とクラスタごとのデータの座標は zip 内の txt ファイルに記入してある。

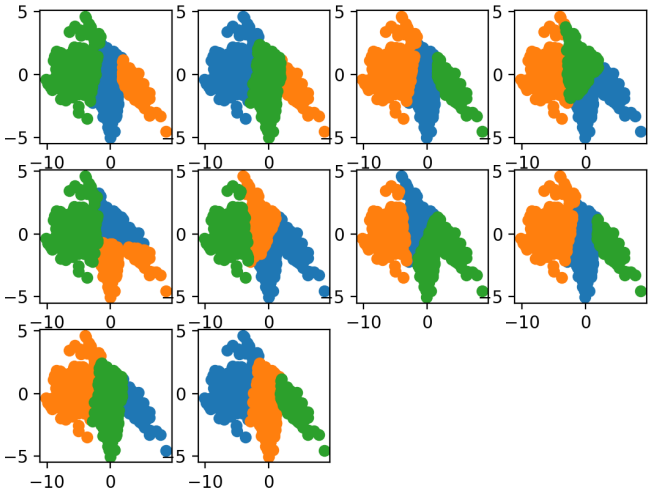

図 19: 課題 21、10 通りのクラスタ分類の結果. 各クラスタごとに色をつけてある.

最も良い分類とその初期値は以下である。

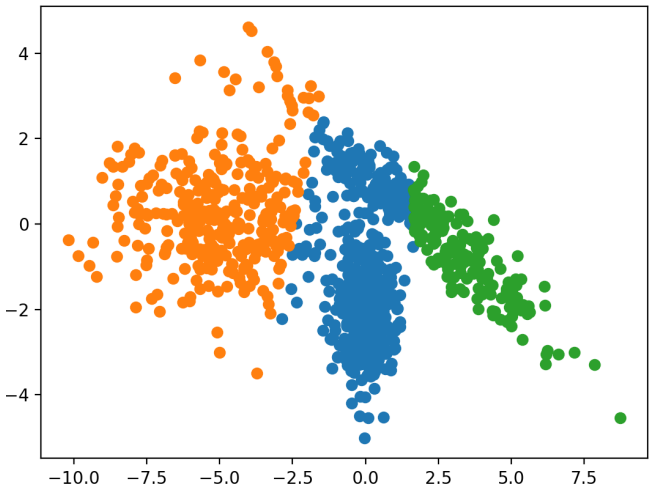

図 20: 課題 21、最も良い分類. 各クラスタごとに色をつけている.

表 21: 最も良い分類を与えた初期値の組.

クラスタ 1 の初期値
(0.21825239843433408 -3.3376625648428835)
クラスタ 2 の初期値
(0.6384426640704373 0.35280648443266394)
クラスタ 3 の初期値
(1.1084801324698716 -4.55197719860286)

ただし、ここで「最も良い分類」とは最終的なクラスタの代表点とそのクラスタに分類されたデータの L2 距離を 1000 個のデータについて平均したものが、10 通りの分類の中で最も小さくなるものと定義した。
なお、kkz 法を用いて初期値を設定した場合のクラスタリングの結果は以下となった。

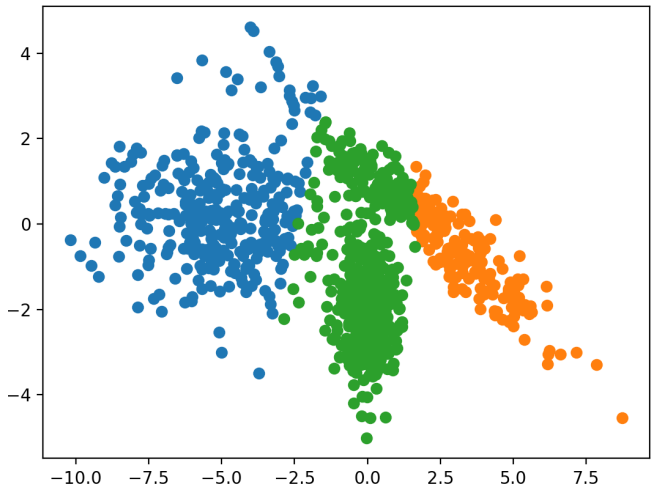

図 21: 課題 21、kkz 法を用いた場合の分類. 各クラスタごとに色をつけている.

ただし、kkz 法のアルゴリズムは以下である。

- データ間距離が最大の 2 データを C_0, C_1 に設定
- 決定したクラスタと各データ x 間の距離 $D(x)$ の和を計算
- 先の手順で求めた和が最大となるデータ x' を次のクラスタとする
- 以上を目的のクラスタ数となるまで繰り返す

58　この課題のまとめと考察

クラスタ分類の結果から、K-means を用いると様々な初期値に対してある程度正確に分類を行えることがわかった。ただ、何度かコードを実行していると、途中でクラスタが消失し最終的に 2 クラス分類となる場合もあったことから、初期値の選択が重要であることも見て取れた。
また、kkz 法を用いた場合とランダムな初期値を用いた場合の最良のクラスタリングとであまり違いは見られなかった (実際、代表点とクラスタ内の距離の平均値は両者で同じだった)。ランダムに初期値を選択する場合でもある程度最終的な代表点に目星をつけていたことが大きな原因であると考えられる。

59 コード

コード 14: 課題 21 のコード

```
import pandas as pd
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt

#データ読み込み
df = pd.read_csv("./suri_jikken6_data/mmse_kadai14.txt",header=
    None)
data = np.array(df)

x_1 = np.array(data[:,0])
x_2 = np.array(data[:,1])
best_dist = 10**6

K = 3
N = 1000
repeat_times = 10
f_ = open("21_meu.txt","w")

#プロットの準備
fig = plt.figure()
ax1 = fig.add_subplot(3, 4, 1)
ax2 = fig.add_subplot(3, 4, 2)
ax3 = fig.add_subplot(3, 4, 3)
ax4 = fig.add_subplot(3, 4, 4)
ax5 = fig.add_subplot(3, 4, 5)
ax6 = fig.add_subplot(3, 4, 6)
ax7 = fig.add_subplot(3, 4, 7)
ax8 = fig.add_subplot(3, 4, 8)
ax9 = fig.add_subplot(3, 4, 9)
ax10 = fig.add_subplot(3, 4, 10)
ax_list = [ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9, ax10]

#10 組みの初期値に対して同様の作業を行う
for time in range(repeat_times):
    x1 = np.random.rand()*20.0 - 10.0
    x2 = np.random.rand()*20.0 - 10.0
    x3 = np.random.rand()*20.0 - 10.0
    y1 = np.random.rand()*10.0 - 5.0
    y2 = np.random.rand()*10.0 - 5.0
    y3 = np.random.rand()*10.0 - 5.0
    meu = [[x1, y1],[x2, y2],[x3 ,y3]]
    meu_tmp = meu
    f_.writelines([str(meu),"\n"])
    meu_pre = [[0.0, 0.0] for _ in range(K)]

    meu = np.array(meu)
    meu_pre = np.array(meu_pre)

    j = 0
    epsilon = 10 ** (-6)

    tmp = np.sum((meu_pre-meu)**2,axis=1)
    max_movement = tmp.max()#パラメータの反復後の変動

    #k-means
    while epsilon <= max_movement:
        r = np.array([[0 for l in range(K)] for i in range(N)])
```

```
58          j += 1
59          #それぞれのデータに対して
60          for i in range(N):
61              l_one = 0
62              dist_min = 100000000
63              x = np.array([x_1[i], x_2[i]])
64              #クラスを決定
65              for l in range(K):
66                  if np.sqrt(np.sum((meu[l]-x)**2)) < dist_min:
67                      l_one = l
68                      dist_min = np.sqrt(np.sum((meu[l]-x)**2))
69              #そのデータに割り振られたクラスの index が 1
70              r[i][l_one] = 1
71          #それぞれのクラスについて
72          for l in range(K):
73              bunbo = np.sum(r, axis=0)[l]
74              bunsi = np.array([0.0, 0.0])
75              for i in range(N):
76                  x = np.array([x_1[i], x_2[i]])
77                  bunsi += r[i][l] * x
78              meu_pre[l] = meu[l]
79              if bunbo != 0:
80                  meu[l] = bunsi / bunbo
81          tmp = np.sum((meu_pre-meu)**2,axis=1)
82          #前の反復からの変動幅
83          max_movement = tmp.max()
84
85      #クラス
86      clusters = np.where(r == 1)
87
88      #クラスタ 0 に分類されたデータ
89      cluster_0 = np.where(clusters[1] == 0)
90      #クラスタ 0 に分類されたデータの第一成分
91      cluster_0_1 = [x_1[i] for i in cluster_0[0]]
92      #クラスタ 0 に分類されたデータの第二成分
93      cluster_0_2 = [x_2[i] for i in cluster_0[0]]
94
95      cluster_1 = np.where(clusters[1] == 1)
96      cluster_1_1 = [x_1[i] for i in cluster_1[0]]
97      cluster_1_2 = [x_2[i] for i in cluster_1[0]]
98
99      cluster_2 = np.where(clusters[1] == 2)
100     cluster_2_1 = [x_1[i] for i in cluster_2[0]]
101     cluster_2_2 = [x_2[i] for i in cluster_2[0]]
102
103     #プロット
104     ax = ax_list[time]
105     ax.scatter(cluster_0_1, cluster_0_2)
106     ax.scatter(cluster_1_1, cluster_1_2)
107     ax.scatter(cluster_2_1, cluster_2_2)
108
109     dist_sum = 0.0 #各データが所属するクラスタの重心からの平均距離
110     a = np.array([[cluster_0_1[i], cluster_0_2[i]] for i in range(len(
            cluster_0_2))])
111     b = np.array(meu[0])
112     if len(cluster_0_2) != 0:
113         dist_sum += np.sum(np.sqrt(np.sum((a - b)**2, axis=1)))
114
115     a = np.array([[cluster_1_1[i], cluster_1_2[i]] for i in range(len(
            cluster_1_2))])
116     b = np.array(meu[1])
117     if len(cluster_1_2) != 0:
```

```
118             dist_sum += np.sum(np.sqrt(np.sum((a - b)**2, axis=1)))
119
120         a = np.array([[cluster_2_1[i], cluster_2_2[i]] for i in range(len(
                cluster_2_2))])
121         b = np.array(meu[2])
122         if len(cluster_2_2) != 0:
123             dist_sum += np.sum(np.sqrt(np.sum((a - b)**2, axis=1)))
124
125         dist_sum = dist_sum / 1000.0
126
127         #今までの初期値の中で最も良い分類だった場合
128         if dist_sum < best_dist:
129             best_cluster_0_1 = cluster_0_1
130             best_cluster_0_2 = cluster_0_2
131             best_cluster_1_1 = cluster_1_1
132             best_cluster_1_2 = cluster_1_2
133             best_cluster_2_1 = cluster_2_1
134             best_cluster_2_2 = cluster_2_2
135             best_dist = dist_sum
136             meu_best = meu_tmp
137
138     #初期値全種類の場合についてプロット
139     plt.show()
140
141     plt.scatter(best_cluster_0_1, best_cluster_0_2)
142     plt.scatter(best_cluster_1_1, best_cluster_1_2)
143     plt.scatter(best_cluster_2_1, best_cluster_2_2)
144
145     #最も良い分類をプロット
146     plt.show()
147
148     #出力
149     f_.writelines(["最も良いクラスタを与えた初期値は",str(meu_best)])
150     f = open("21_clusters.txt","w")
151     f.write("各データ点の、所属するクラスタの重心からの距離の平均は以下
            ␣\n")
152     f.write(str(best_dist))
153     f.write("\n")
154     f.write("cluster0␣は以下の点␣\n")
155     f.writelines([str((best_cluster_0_1[i], best_cluster_0_2[i])) for i in range(len(
            best_cluster_0_2))])
156     f.write("\n")
157     f.write("cluster1␣は以下の点␣\n")
158     f.writelines([str((best_cluster_1_1[i], best_cluster_1_2[i])) for i in range(len(
            best_cluster_1_2))])
159     f.write("\n")
160     f.write("cluster2␣は以下の点␣\n")
161     f.writelines([str((best_cluster_2_1[i], best_cluster_2_2[i])) for i in range(len(
            best_cluster_2_2))])
162
163     f.close()
164     f_.close()
```

第 XVIII 部
レポートのまとめ

このレポートでは、最小二乗法の関係する様々な問に回答し、理解を深めた。重回帰問題、多項式回帰問題を解き (課題 8、9)、観測誤差の分布やデータの取り方によって結果にどのような違いが出るかを確認し、その理由を考察した (課題 10、11)。観測値の次元が多次元になる場合についても同様のことを行い (課題 12)、観測誤差の分布が異なるデータを組み合わせたものに対しても回帰を行った (課題 13)。データを分割してパラメータを推定し、それらを合成することで、全データを使用した場合と同様の推定値を得られることを確認した (課題 14)。さらに偶然誤差の分散の推定値を求め、それを用いてパラメータの推定値を合成し、全データを用いた回帰により得られたパラメータとどちらが優れているか比較、考察した。結果、合成して得た推定値の方が優れていることがわかった (課題 15)。バネマスダンパ系を表す直線状の運動方程式のパラメータを逐次最小二乗法で求めた (課題 16)。この際、正則化項などに工夫を加えた。正弦波で変動する信号から得られたデータに対して忘却係数つきの逐次最小二乗法で回帰を行い、各時刻でパラメータを推定した (課題 17)。Kalman フィルタを実装し、与えられた離散時間ダイナミクスおよび観測方程式に適用することで時間ごとのパラメータの推定値を求めた (課題 18)。RTS アルゴリズムを実装し、課題 18 の設定のもとでパラメータの初期値を推定し、またその推定誤差分散を求めた (課題 19)。交互最小二乗法を実装し、与えられたデータに適用することでパラメータを推定した。また、初期値を変えて同様のことを繰り返し行った (課題 20)。最後に、与えられたデータに k-means 法を適用し、3 つのクラスターに分類した。異なる初期値に対して同様のことをおこなった。結果、k-means 法は初期値により影響を受けるものの、それなりの精度でクラスタリングを行えることがわかった。