センシング工学特論
レポート課題【課題 ＩＩ】

3G150131　松井 健人

2015 年 6 月 17 日

1　課題 Ⅰ

観測方程式

$$z = x_0 + x_1 a + x_2 a^2 = \begin{bmatrix} 1 & a & a^2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + w = \boldsymbol{a}^T \boldsymbol{x} + w \tag{1}$$

に関して下表の観測値が得られた．

表 1: C 言語の代表的なデータの型

a	z	a	z	a	z	a	z
-15	162.1746	-8	26.5951	0	-1.9845	8	93.7607
-14	139.5805	-7	20.1832	1	2.0593	9	112.6638
-13	113.8133	-6	8.8816	2	12.3849	10	134.9818
-12	94.3372	-5	1.8636	3	17.9044	11	162.7143
-11	74.7258	-4	-5.0213	4	30.1826	12	188.9610
-10	59.3817	-3	-5.8861	5	41.1677	13	219.6236
-9	41.4117	-2	-5.7711	6	55.7128	14	248.9036
		-1	-4.9332	7	74.2944	15	281.3082

なお，a が奇数のときの観測雑音 w は平均値 0，分散 1 の正規分布であり，a が偶数のときの観測雑音の w は平均値 0，分散 4 の正規分布である．

(1) カルマンフィルタにより未知数（一定値）x を推定するアルゴリズムを示せ．また，プログラムを作成して推定値を求めよ．なお，プログラミング言語としては何でも良いがカルマンフィルタの関数は使わないこと．

(2) プラント雑音の共分散，推定値の初期共分散に対する推定精度への影響を調べよ。

ただし，x の解は $\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \\ 1 \end{bmatrix}$ である．

2　カルマンフィルタによる未知数の推定

2.1　カルマンフィルタ

カルマンフィルタとは，観測雑音とプラント雑音により誤差が生じた過去の観測値を基に，観測方程式と状態方程式を用いることで現在の観測値を推定する手法である．未知数の推定方法は大きく分けて、先験情報と観測値から推定するベイズ推定と観測値のみにより推定するフィッシャー推定の 2 つに分類され，カルマンフィルタはベイズ推定に分類される．

2.2　カルマンフィルタによる推定アルゴリズム

カルマンフィルタは，任意の時刻における未知数を推定するために，その時刻の 1 つ前の時刻において推定した数値を補正する方法である．この方法では，任意の時刻の 1 つ前の時刻の推定値が必要となるが，最初の時刻では，1 つ前に推定値が存在しないため推定することができない．そこで，推定値を 0 とし，推定誤差共分散を非常に大きい値とする．ここで，推定誤差共分散が非常に大きい値であることは，その推定値が信用出来ないことを意味する．

カルマンフィルタでは，推定値が時間に従って変化するので観測値を扱うので観測方程式 (1) を式 (2) のように変換する．また，状態方程式を式（3）に示す．

$$\text{観測方程式：}\quad \boldsymbol{z}_{(k)} = \boldsymbol{H}_{(k)}\boldsymbol{x}_{(k)} + \boldsymbol{w}_{(k)} \tag{2}$$

$$\text{状態方程式：}\quad \boldsymbol{x}_{(k+1)} = \boldsymbol{F}_{(k)}\boldsymbol{x}_{(k)} + \boldsymbol{G}_{(k)}\boldsymbol{u}_{(k)} + \boldsymbol{v}_{(k)} \tag{3}$$

また，観測雑音 $\boldsymbol{w}^{(k)}$ の共分散 $\boldsymbol{R}^{(k)}$，プラント雑音 $\boldsymbol{v}^{(k)}$ の共分散 $\boldsymbol{Q}^{(k)}$ とする．ここで，プラント雑音共分散 $\boldsymbol{Q}^{(k)}$ はサンプリング周期 τ によって表すことができる．

カルマンフィルタでは，式（2）に示す観測方程式，観測雑音の共分散，式（3）に示す状態方程式，プラント雑音の共分散，前ステップの推定値および推定誤差共分散を基に，予測値，予測誤差共分散，観測予測値，観測予測誤差共分散，観測予測誤差共分散，フィルタゲインをそれぞれ式（4）-（9）を用いて求める．

$$\text{予測値：}\quad \hat{\boldsymbol{x}}_{(k+1|k)} = \boldsymbol{F}_{(k)}\hat{\boldsymbol{x}}_{(k|k)} + \boldsymbol{G}_{(k)}\boldsymbol{u}_{(k)} \tag{4}$$

$$\text{予測誤差共分散：}\quad \boldsymbol{P}_{(k+1|k)} = \boldsymbol{F}_{(k)}\boldsymbol{P}_{(k|k)}{\boldsymbol{F}^T}_{(k)} + \boldsymbol{Q}_{(k)} \tag{5}$$

$$\text{観測予測値：}\quad \hat{\boldsymbol{z}}_{(k+1|k)} = \boldsymbol{H}_{(k+1)}\hat{\boldsymbol{x}}_{(k+1|k)} \tag{6}$$

$$\text{観測予測誤差：}\quad \tilde{\boldsymbol{z}}_{(k+1|k)} = \boldsymbol{z}_{(k+1)} - \hat{\boldsymbol{z}}_{(k+1|k)} \tag{7}$$

$$\text{観測予測誤差共分散：}\quad \boldsymbol{S}_{(k+1)} = \boldsymbol{H}_{(k+1)}\boldsymbol{P}_{(k+1|k)}{\boldsymbol{H}^T}_{(k+1)} + \boldsymbol{R}_{k+1} \tag{8}$$

$$\text{フィルタゲイン：}\quad \boldsymbol{W}_{(k+1)} = \boldsymbol{P}_{(k+1|k)}{\boldsymbol{H}^T}_{(k+1)}{\boldsymbol{S}^{-1}}_{(k+1)} \tag{9}$$

推定値および推定誤差共分散は式 (4)- (9) よりそれぞれ式 (10) および (11) と表すことができる．

$$\text{推定値：}\quad \hat{\boldsymbol{x}}_{(k+1|k+1)} = \hat{\boldsymbol{x}}_{(k+1|k)} + \boldsymbol{W}_{(k+1)}\tilde{\boldsymbol{z}}_{(k+1|k)} \tag{10}$$

$$\text{推定誤差共分散：}\quad \boldsymbol{P}_{(k+1|k+1)} = \boldsymbol{P}_{(k+1|k)} - \boldsymbol{W}_{(k+1)}\boldsymbol{S}_{(k+1)}{\boldsymbol{W}^T}_{(k+1)} \tag{11}$$

カルマンフィルタでは未知数を式 (10) を用いて推定する．

2.2.1 カルマンフィルタにより推定するプログラム

カルマンフィルタにより未知数を推定するプログラムをオブジェクト指向スクリプト言語 Ruby で作成した．カルマンフィルタでは，任意の時刻の 1 つ前の時刻の推定値が必要となるが，最初の時刻では，1 つ前に推定値が存在しないため推定することができない．そこで，初期推定値を 0 とし，初期推定誤差共分散を適当に設定して推定精度の検証を行う．このとき初期推定誤差共分散は，$\boldsymbol{P}_{(0)} = \begin{bmatrix} 10000 & 0 & 0 \\ 0 & 10000 & 0 \\ 0 & 0 & 10000 \end{bmatrix}$ とする．また，プラント雑音の共分散による影響はないものとする．

カルマンフィルタにより未知数を推定するプログラムのソースコードをリスト 1 に示す．また，出力結果をリスト 2 に示し，推定値の時系列グラフを図 1 に示す．

リスト 1: バッチ型最小二乗法により推定するプログラムのソースコード

```
1   # coding:utf-8
2
3   require 'matrix'
4   require 'csv'
5
6   #カルマンフィルタを用いて未知数 x を推定するアルゴリズム
7   class Kalman
8           #観測値, 観測雑音の分散, プラント雑音の共分散, 推定値の初期値, 推定誤差共分散の初期値を設定
9           def initialize(z, r, q, x, p)
10                  @z = z
11                  @r = r
12                  @q = q
13                  @x = x
14                  @p = p
15          end
16
17          #推定値の計算
18          #num: 現在の時間
19          #return: 計算した推定値
20          def estimate(num)
21                  k = num - 15
22                  #状態方程式の変数
23                  f = Matrix.unit(3)
24                  #予測値
25                  pre_next_x = f * @x
26                  #予測誤差共分散
27                  pre_next_p = f * @p * f.transpose + @q
28
29                  #観測雑音の共分散
30                  next_r = Matrix[[@r[((k + 1) + 1) % 2]]]
31                  #観測方程式の変数：a(1, a, a^2)
32                  next_h = Matrix[[1, (k + 1), (k + 1) ** 2]]
33                  #観測予測誤差共分散
34                  next_s =  next_h * pre_next_p * next_h.transpose + next_r
35                  #フィルタゲイン
36                  next_w = pre_next_p * next_h.transpose * next_s.inverse
37                  #観測予測値
38                  pre_next_z = next_h * pre_next_x
39                  #観測予測誤差
40                  err_pre_next_z = Matrix[[@z[num + 1]]] - pre_next_z
41                  #推定値   : x(x0, x1, x2)
42                  est_next_x = pre_next_x + next_w * err_pre_next_z
43                  #推定誤差共分散
44                  est_next_p = pre_next_p - next_w * next_s * next_w.transpose
```

```
45
46                  #使用する観測値を抽出
47                  @x = est_next_x
48                  @p = est_next_p
49
50                  #推定値を返す
51                  return @x
52          end
53  end
54
55  #観測値
56  z = [162.1746, 139.5805, 113.8133,  94.3372,  74.7258,  59.3817,  41.4117,  26.5951,
57        20.1832,   8.8816,   1.8636,  -5.0213,  -5.8861,  -5.7711,  -4.9332,  -1.9845,
58         2.0593,  12.3849,  17.9044,  30.1826,  41.1677,  55.7128,  74.2944,  93.7607,
59       112.6638, 134.9818, 162.7143, 188.9610, 219.6236, 248.9036, 281.3082]
60
61  #観測雑音の分散
62  w = [1, 4]
63
64  #サンプリング周期
65  t = 0
66
67  #プラント雑音の共分散
68  q = Matrix[[(t ** 5) / 20, (t ** 4) / 8, (t ** 3) / 6],
69             [(t ** 4) /  8, (t ** 3) / 3, (t ** 2) / 2],
70             [(t ** 3) /  6, (t ** 2) / 2,            t]]
71
72  #推定値の初期値
73  init_x = Matrix[[0], [0], [0]]
74
75  #推定誤差共分散の初期値
76  init_p = Matrix.scalar(3, 1000)
77
78  main = Kalman.new(z, w, q, init_x, init_p)
79
80  CSV.open("./Kalman3-1[p=#{init_p[0, 0]}_t=#{t}].csv", "w") do |write|
81          write << ["a", "x0", "x1", "x2"]
82          write << ["-15", 0, 0, 0]
83
84          (z.size - 1).times do |i|
85                  x = main.estimate(i)
86                  write << [(i + 1) - 15, x[0, 0], x[1, 0], x[2, 0]]
87          end
88  end
89
```

リスト 2: バッチ型最小二乗法により推定するプログラムの出力結果

```
1   a       x0                      x1                      x2
2   -15     0                       0                       0
3   -14     0.0036148576529680923   -0.05060800714155329    0.708512099981746
4   -13     -1.0282729049119361     6.886533148043633       1.2092736967942677
5   -12     35.25960539855118       10.534882019483499      1.278858229612447
6   -11     45.27726709886268       11.894850079054 1.3243033518327476
7   -10     54.924066014805945      13.433164781390854      1.3852792845964506
8   -9      -14.19077927229057      2.096712512265496       0.9242295568901613
9   -8      -23.610594861788613     0.4512905939951497      0.8535733433449237
10  -7      10.643318817358619      6.6003951106139755      1.1236449033381646
11  -6      4.770402334889124       5.478313430891044       1.0718817852673483
12  -5      -2.7381389137887053     3.9927382190005405      1.001530542444141
```

```
13  -4  -5.032646344227069  3.5059536971945127  0.977307013335598
14  -3  -3.584514366353549  3.8274352743254254  0.9937971088871869
15  -2  -2.8868347817064572  3.995700335804556  1.0028856970733764
16  -1  -2.0537194501256684  4.2090676672990694  1.0148163673956065
17  0  -2.037728549062292  4.213613433770274  1.0150848857847778
18  1  -2.5617413683280494  4.05175232710843  1.005140459963322
19  2  -2.2424197020446064  4.165428752681028  1.0125502838824298
20  3  -2.5811844294215485  4.028836247621476  1.0032401459328246
21  4  -2.550780863703024  4.044094160162502  1.0043497118360902
22  5  -2.710934798473771  3.943941379255451  0.9966854385595484
23  6  -2.7300412130173943  3.925328527370832  0.9951547024316857
24  7  -2.7122430881063924  3.958149828356845  0.9980175773931789
25  8  -2.713843590294599  3.9683512655615227  0.9989830533174435
26  9  -2.686372227090556  3.9329578194405017  0.9953941899811616
27  10  -2.671945636791899  3.9243362079896125  0.9944332423304582
28  11  -2.761411031164477  3.960000859933322  0.9987509852203267
29  12  -2.768080254302869  3.9617846833066306  0.9989927965829036
30  13  -2.8937021862504655  3.9871450895280116  1.0028016086265994
31  14  -2.8822520995831904  3.985463194498891  1.0025106831872368
32  15  -2.8041031361411672  3.9765704645732387  1.0007521605967222
```

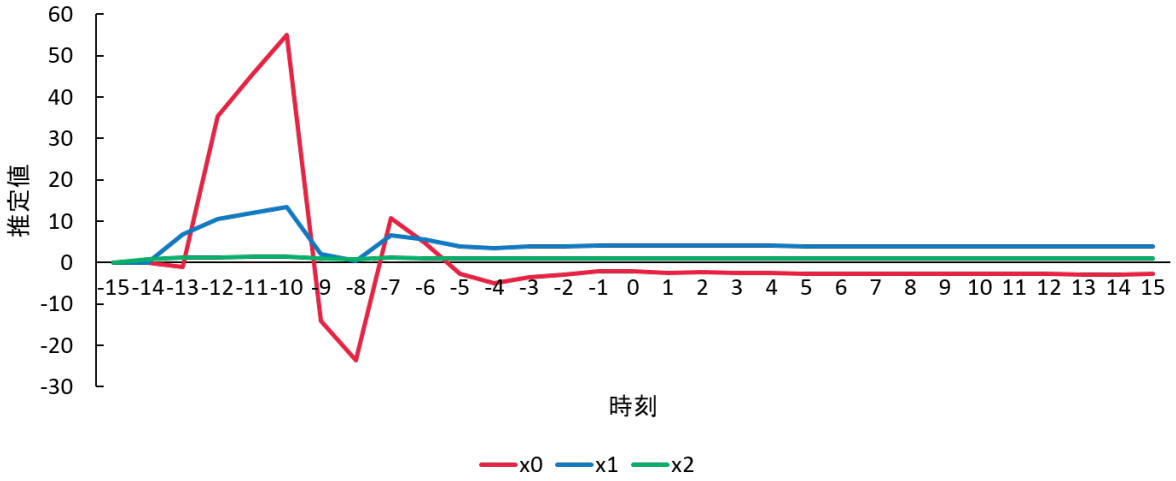

図 1: カルマンフィルタによる推定値の時系列グラフ

リスト 2 より，時刻 a が 15 のとき推定値は $x_0 = -2.804, x_1 = 3.977 x_2 = 1.001$ となっており，正しく推定できていることがわかる．また，図 1 より，推定値が収束していることがわかる．

3　カルマンフィルタにおける推定精度への影響

3.1　カルマンフィルタにおけるプラント雑音の共分散に対する推定精度への影響

カルマンフィルタにおいて，プラント雑音の共分散に対する推定精度への影響を検証する．ここで，サンプリング周期を τ とするときプラント雑音の共分散は $\boldsymbol{Q}_{(k)} = \begin{bmatrix} \frac{\tau^5}{20} & \frac{\tau^4}{8} & \frac{\tau^3}{6} \\ \frac{\tau^4}{8} & \frac{\tau^3}{3} & \frac{\tau^2}{2} \\ \frac{\tau^3}{6} & \frac{\tau^2}{2} & \tau \end{bmatrix} q$ とする．また，初期推定誤差共分散は，$\boldsymbol{P}_{(0)} = \begin{bmatrix} 10000 & 0 & 0 \\ 0 & 10000 & 0 \\ 0 & 0 & 10000 \end{bmatrix}$ とする．

$\tau = 0, 10^{-6}, 10^{-3}, 10^{0}, 10^{3}, 10^{6}$ の 6 つの場合における，カルマンフィルタによる推定値の時系列グラフを図 2 に示す．

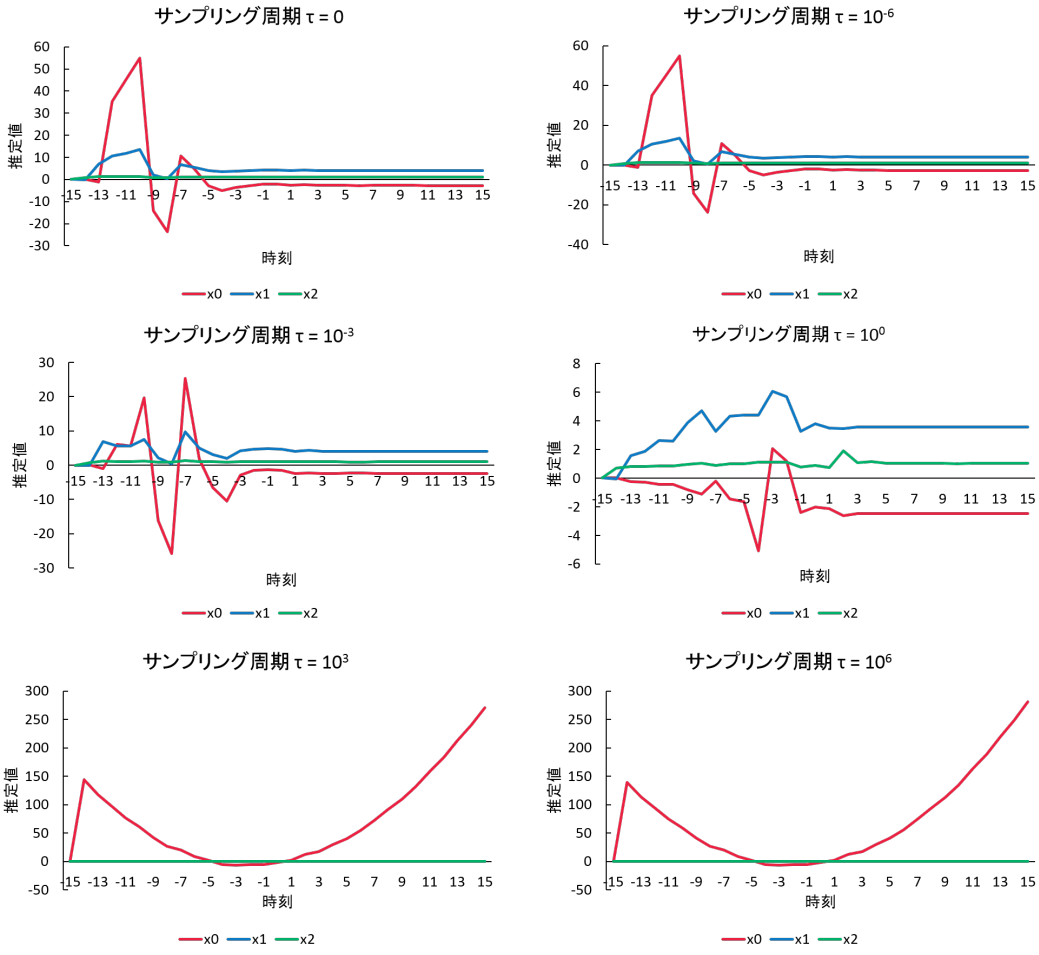

図 2: サンプリング周期 τ を変化させた際の推定値の時系列グラフ

図 2 より，サンプリング周期 τ の値を 0 から徐々に大きくすると，推定値の誤差の振れ幅は小さくなるものの，収束までにかかる時間が長くなることがわかる．また，サンプリング周期 τ が大きくなりすぎると推定値が発散してしまい，推定できないことがわかった．

3.2　カルマンフィルタにおける推定値の初期共分散に対する推定精度への影響

カルマンフィルタにおいて，推定値の初期共分散に対する推定精度への影響を検証する．このとき推定の初期共分散は，$\boldsymbol{P}_{(0)} = \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{bmatrix}$ とする．また，サンプリング周期は $\tau = 10^0$ とする．

$p = 0, 10^{-6}, 10^{-3}, 10^0, 10^3, 10^6$ の 6 つの場合における，カルマンフィルタによる推定値の時系列グラフを図 3 に示す．

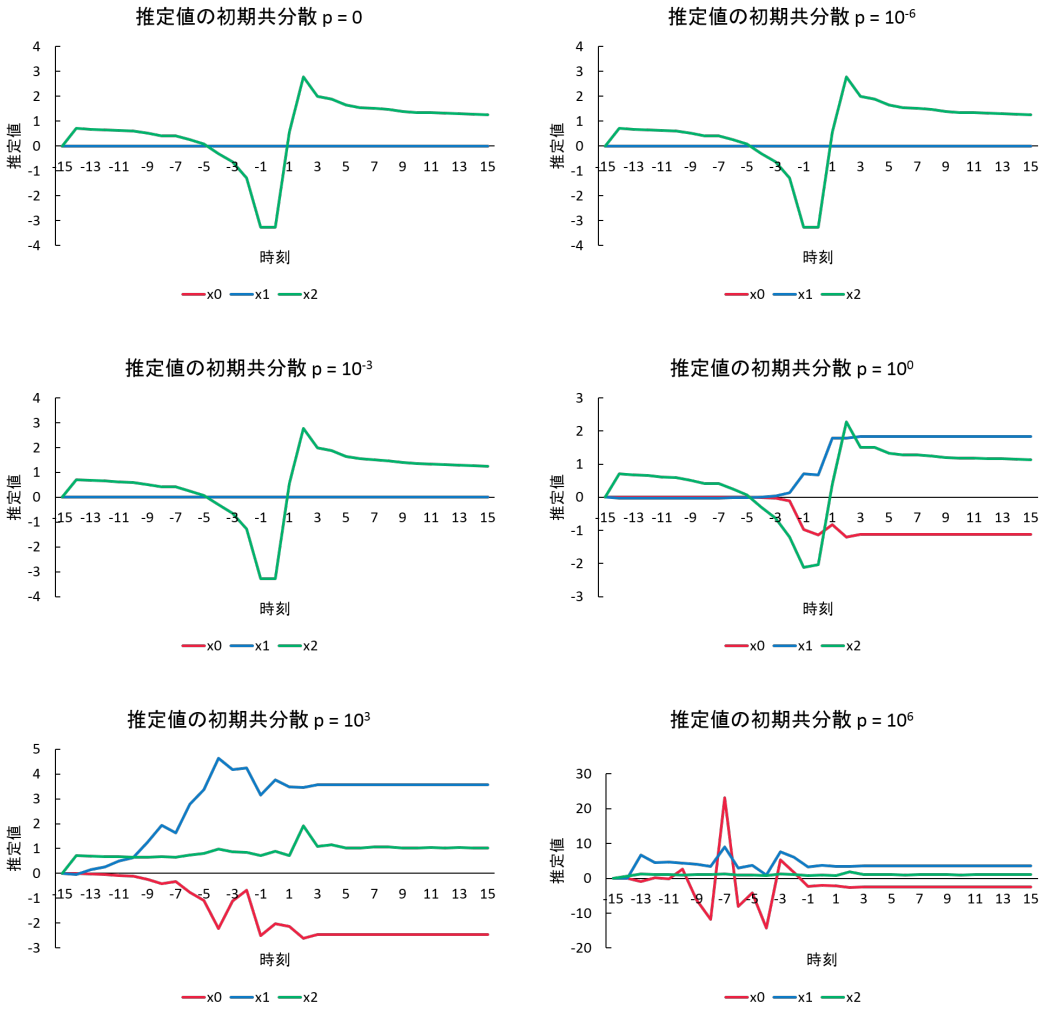

図 3: 推定値の初期共分散 p を変化させた際の推定値の時系列グラフ

図 3 より，サンプリング周期 $\tau = 10^0$ の場合初期推定誤差共分散の値が小さすぎると，推定値 x_0, x_1, x_2 の値がおおよそ同じ値となり，時系列グラフの形状が変化しないことがわかった．一方，初期推定誤差共分散の値が大きなると，収束するので推定が可能であることがわかる．これより，初期推定誤差共分散を非常に大きくすることで推定誤差が小さくなることがわかる．これは，推定誤差共分散が小さいということはその値が真値に近いことを示しているためあまり補正されないが，推定誤差共分散が大きいと真値から離れていることを示すので大きく補正がされることが原因であると考えられる．