センシング工学特論
レポート課題【課題 Ⅰ 】

3G150131　松井 健人

2015 年 6 月 5 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) バッチ型最小二乗法，逐次型最小二乗法それぞれにより未知数（一定値）x を推定するアルゴリズムを示せ．また，実際にプログラムを作成して推定値を求めよ．プログラミング言語としては何を用いてもよいが最小二乗法の関数は使わないこと．

(2) 逐次型最小二乗法において，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　最小二乗法による未知数の推定

2.1　最小二乗法

最小二乗法とは，観測により得た数値を 1 次関数や対数曲線などの関数を用いて近似する際に，誤差の二乗和が最小となる係数を決定することで未知数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，最小二乗法はフィッシャー推定に分類される．また，最小二乗法にはバッチ型最小二乗法と逐次型最小二乗法がある．

2.2　バッチ型最小二乗法

2.2.1　バッチ型最小二乗法による推定アルゴリズム

バッチ型最小二乗法とは，任意の時刻における未知数を推定するために，それ以前に観測された数値を一括で処理する方法である．この方法では，多くのデータを扱うこととなるため，計算時間が増大し，大きなメモリ容量が必要となる．

バッチ型最小二乗法では，複数の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．

$$\begin{bmatrix} z_{(1)} \\ \vdots \\ z_{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{(1)} \\ \vdots \\ \boldsymbol{H}_{(k)} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} w_{(1)} \\ \vdots \\ w_{(k)} \end{bmatrix} \iff \boldsymbol{z}^k = \boldsymbol{H}^k \boldsymbol{x} + \boldsymbol{w}^k \tag{2}$$

また，観測雑音 $\boldsymbol{w}^k$ の共分散 $\boldsymbol{R}^k$ を式（3）に示す．

$$\boldsymbol{R}^k = \begin{bmatrix} \boldsymbol{R}_{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{R}_{(k)} \end{bmatrix} \tag{3}$$

推定値および推定誤差共分散は式 (2), (3) よりそれぞれ式 (4) および (5) と表すことができる．

$$\hat{\boldsymbol{x}}_{(k)} = \left[(\boldsymbol{H}^k)^T (\boldsymbol{R}^k)^{-1} \boldsymbol{H}^T \right]^{-1} (\boldsymbol{H}^k)^T (\boldsymbol{R}^k)^{-1} \boldsymbol{z}^k \tag{4}$$

$$\boldsymbol{P}_{(k)} = \left[(\boldsymbol{H}^k)^T (\boldsymbol{R}^k)^{-1} \boldsymbol{H}^T \right]^{-1} \tag{5}$$

バッチ型最小二乗法では未知数を式 (4) を用いて推定する．

2.2.2 バッチ型最小二乗法により推定するプログラム

バッチ型最小二乗法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．今回は時刻 a が-15 から 15 までを想定して各時刻における未知数の推定を行うが，項が 3 つ存在するので過去の観測データが 3 つ必要なる．そのため，時刻 a が-13 となってから推定を開始する．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．時刻 a が-13，-12 のとき推定値の誤差が大きいため，それらの時刻を除いた推定値の時系列グラフを図 2 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  #バッチ型最小二乗法を用いて未知数 x を推定するアルゴリズム
7  class Batch
8          #観測値と観測雑音を設定
9          def initialize(z, w)
10                 @z = z
11                 @w = w
12         end
13
14         #推定値の計算
15         #num: 現在の時間
16         #return: 計算した推定値
17         def estimate(num)
18                 #観測雑音の共分散行列
19                 r = Array.new(num){Array.new(num, 0)}
20                 #変数行列：a(1, a, a^2)
21                 a = Array.new(num){Array.new(3, 0)}
22                 #推定値   ：x(x0, x1, x2)
23                 x = Array.new(3, 0)
24
25                 #変数行列と観測雑音の共分散行列を設定
26                 num.times do |i|
27                         k = i - 15
28                         a[i][0] = 1
29                         a[i][1] = k
30                         a[i][2] = k ** 2
31                         r[i][i] = @w[(k + 1) % 2]
32                 end
33
34                 #配列を行列に変換
35                 a = Matrix.rows(a)
36                 r = Matrix.rows(r)
37                 #使用する観測値を抽出
38                 z = Matrix.rows([@z.slice(0, num)])
39                 #推定値を計算
40                 x = (a.transpose * r.inverse * a).inverse * a.transpose * r.inverse * z.transpose
41                 #推定誤差共分散
42                 p = (a.transpose * r.inverse * a).inverse
43           puts p
44                 #推定値を返す
45                 return x
46         end
47 end
48
49 #観測値
```

```
z = [ 162.1746, 139.5805, 113.8133,  94.3372,  74.7258,  59.3817,  41.4117,  26.5951,
       20.1832,   8.8816,   1.8636,  -5.0213,  -5.8861,  -5.7711,  -4.9332,  -1.9845,
        2.0593,  12.3849,  17.9044,  30.1826,  41.1677,  55.7128,  74.2944,  93.7607,
      112.6638, 134.9818, 162.7143, 188.9610, 219.6236, 248.9036, 281.3082]

#観測雑音
w = [1.0, 4.0]

#バッチ型最小二乗法を用いて未知数 x を推定
main = Batch.new(z, w)

#CSV ファイルへの書き出し
CSV.open("/home/kent/ruby/BatchDatas.csv", "w") do |write|
        write << ["a", "x0", "x1", "x2"]
        2.upto(z.size-1) do |i|
                x = main.estimate(i)
                write << [(i) - 15, x[0, 0], x[1, 0], x[2, 0]]
        end
end
```

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

```
a       x0                      x1                      x2
-13     60.50115000000005       -2.4319390624999997     1.2836657226562505
-12     -509.9123999735075      -68.60404999752814      -1.5865499999019193
-11     9.861463999326588       5.784561999784046       1.0634259999913667
-10     5.285103755923046       5.12186502474367        1.0395668674687393
-9      19.978120252291546      7.353631545705824       1.123351861197539
-8      -15.56737422880849      1.8457200655088144      0.9130330330326544
-7      -21.117303526544262     0.9411012318885668      0.8770228743961785
-6      3.661629595959326       5.090778333333409       1.0457896464646277
-5      0.6858849152655946      4.5639565819225965      1.023439632768459
-4      -3.4839087499960115     3.799725295542949       0.9901921085271401
-3      -5.149765052430663      3.474460433967193       0.9754103582074325
-2      -3.8461851749190217     3.7404195251687007      0.9878477599782807
-1      -3.21377759928348       3.8794811620427883      0.9946552240183529
0       -2.2781422327352416     4.097198418032143       1.0056612053361933
1       -2.216681041193295      4.112891133900216       1.006493925434328
2       -2.5945756902968844     4.0088266119131015      1.0007697706584329
3       -2.267969265929523      4.11047243845282        1.0066593926804748
4       -2.561840461557777      4.0086108792806785      1.0005151266921923
5       -2.5232757651535054     4.024572171392137       1.001533777832752
6       -2.6953962520447137     3.939896371432501       0.995880034272436
7       -2.7202525091509875     3.923258814798968       0.9946981919189385
8       -2.686496095414006      3.955830854827342       0.9971333473226258
9       -2.6823708568410014     3.9661684758207296      0.997961366139611
10      -2.676955859964383      3.93290145121938        0.9951360723329656
11      -2.6688886154855886     3.9243592087407126      0.9943517153839841
12      -2.7307369661953715     3.9616072262226014      0.9980128871417306
13      -2.7365052614398793     3.963772340190573       0.9982457984888897
14      -2.8409817479651043     3.9926022537900314      1.0016086779154245
15      -2.833216701014643      3.991047482902261       1.0014067670583455
```

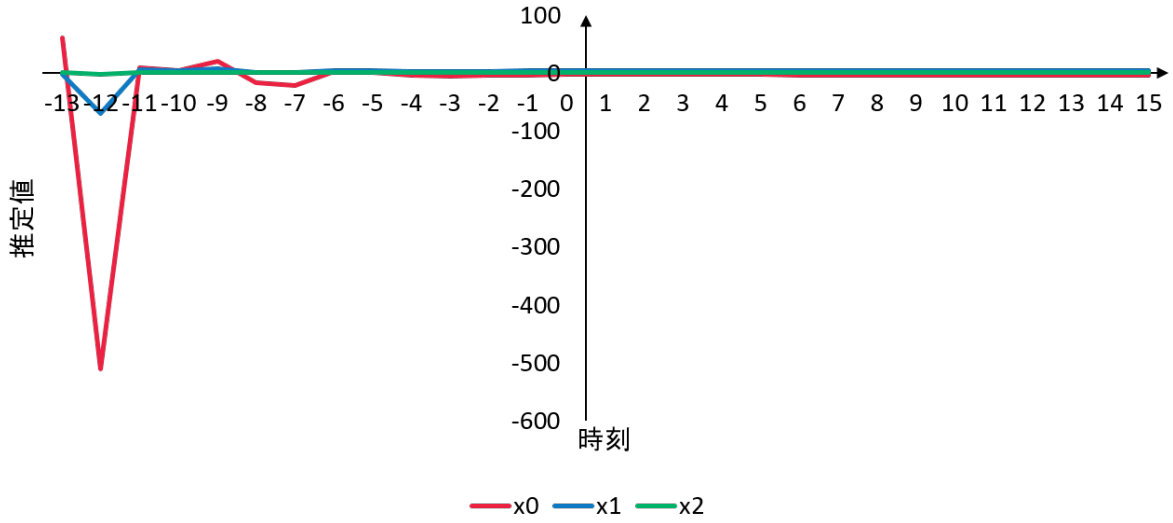

図 1: バッチ型最小二乗法による推定値の時系列グラフ (-13≤ a ≤15)

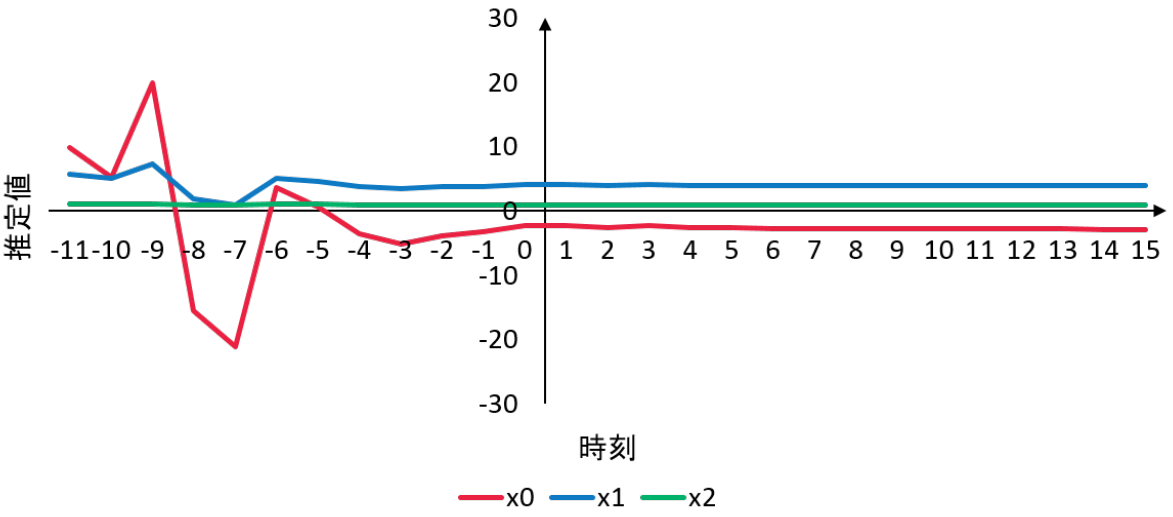

図 2: バッチ型最小二乗法による推定値の時系列グラフ (-11≤ a ≤15)

リスト 2 より，時刻 a が 15 のとき推定値は $x_0 = -2.833, x_1 = 3.991 x_2 = 1.001$　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 1,2 より，推定値が収束していることがわかる．

2.3 逐次型最小二乗法

2.3.1 逐次型最小二乗法による推定アルゴリズム

逐次型最小二乗法とは，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，最初の数サイクルにおいてバッチ型最小二乗法を用いて推定値および推定誤差共分散を導出する，あるいは，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必要がある．ここで，推定誤差共分散が非常に大きい場合，その推定値は信用出来ないことを意味する．

バッチ型最小二乗法では，複数の観測値を扱うので観測方程式 (1) および観測雑音の共分散，推定誤差共分散により観測予測誤差共分散を求める．推定誤差共分散からフィルタゲインを求める．観測予測誤差共分散およびフィルタゲインを式 (6), (7) に示す．

$$\boldsymbol{S}_{(k+1)} = \boldsymbol{H}_{(k+1)}\boldsymbol{P}_{(k)}{\boldsymbol{H}^T}_{(k+1)} + \boldsymbol{R}_{k+1} \tag{6}$$

$$\boldsymbol{W}_{(k+1)} = \boldsymbol{P}_{(k)}{\boldsymbol{H}^T}_{(k+1)}{\boldsymbol{S}^{-1}}_{(k+1)} \tag{7}$$

推定値および推定誤差共分散は式 (6), (7) よりそれぞれ式 (8) および (9) と表すことができる．

$$\hat{\boldsymbol{x}}_{(k+1)} = \hat{\boldsymbol{x}}_{(k)} + \boldsymbol{W}_{(k+1)}\left[z_{(k+1)} - \boldsymbol{H}_{(k+1)}\hat{\boldsymbol{x}}_{(k)}\right] \tag{8}$$

$$\boldsymbol{P}_{(k+1)} = \boldsymbol{P}_{(k+1)} - \boldsymbol{W}_{(k+1)}\boldsymbol{S}_{(k+1)}{\boldsymbol{W}^T}_{(k+1)} \tag{9}$$

逐次型最小二乗法では未知数を式 (8) を用いて推定する．

2.3.2 逐次型最小二乗法により推定するプログラム

逐次型最小二乗法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．今回は時刻 a が-15 から 15 までを想定して各時刻における未知数の推定を行う．最初の推定値および推定誤差共分散は，バッチ型最小二乗法を用いて時刻 a が-15 から-13 までに推定する．その値を用いて，時刻 a が-12 のときから逐次型最小二乗法により未知数を推定する．時刻 a が-12 のときの推定値は $\hat{\boldsymbol{x}}_{(-12)} = \begin{bmatrix} 60.50 \\ -2.432 \\ 1.284 \end{bmatrix}$，$\boldsymbol{P}_{(-12)} = \begin{bmatrix} -3.137 \times 10^{15} & -4.332 \times 10^{14} & -1.494 \times 10^{13} \\ -4.332 \times 10^{14} & -5.982 \times 10^{13} & -2.063 \times 10^{12} \\ -1.494 \times 10^{13} & -2.063 \times 10^{12} & -7.113 \times 10^{10} \end{bmatrix}$ である．ソースコードをリスト 3 に示す．また，出力結果をリスト 4 に示す．

リスト 3: 逐次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  #逐次型最小二乗法を用いて未知数 x を推定するアルゴリズム
7  class Sequential
8          #観測値，観測雑音，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, w, x, p)
10                 @z = z
11                 @w = w
12                 @x = x
13                 @p = p
14         end
15
16         #推定値の計算
17         #num: 現在の時間
18         #return: 計算した推定値
19         def estimatex(num)
20                 k = num - 15
21                 #観測雑音の共分散
22                 next_r = Matrix[[@w[((k + 1) + 1) % 2]]]
23                 #puts (k + 1), next_r
24                 #変数行列：a(1, a, a^2)
25                 next_a = Matrix[[1, (k + 1), (k + 1) ** 2]]
26                 #観測予測誤差共分散
27                 next_s =  next_a * @p * next_a.transpose + next_r
28                 #フィルタゲイン
29                 next_w = @p * next_a.transpose * next_s.inverse
30                 #推定値  ：x(x0, x1, x2)
31                 next_x = @x + next_w * (Matrix[[@z[num + 1]]] - next_a * @x)
32
33                 #推定誤差共分散
34                 next_p = @p - next_w * next_s * next_w.transpose
35
36                 #使用する観測値を抽出
37                 @x = next_x
38                 @p = next_p
39
40                 #推定値を返す
41                 return next_x
42         end
43 end
44
45 #観測値
46 z = [ 162.1746, 139.5805, 113.8133,  94.3372,  74.7258,  59.3817,  41.4117,  26.5951,
47        20.1832,   8.8816,   1.8636,  -5.0213,  -5.8861,  -5.7711,  -4.9332,  -1.9845,
48         2.0593,  12.3849,  17.9044,  30.1826,  41.1677,  55.7128,  74.2944,  93.7607,
49       112.6638, 134.9818, 162.7143, 188.9610, 219.6236, 248.9036, 281.3082]
50
51 #観測雑音
52 w = [1.0, 4.0]
53
54 #推定値の初期値
55 init_x = Matrix[[60.5011500000001], [-2.4319390625], [1.28366572265625]]
56
57 #推定誤差共分散の初期値
58 init_p = Matrix[[-3136835561352321.5, -433182053710581.75,  -14937312196917.37],
59                 [-433182053710577.25,   -59820378845744.1, -2062771684335.9282],
```

```
60                  [-14937312196917.068,  -2062771684335.908,  -71130058080.54001]]
61
62  main = Sequential.new(z, w, init_x, init_p)
63
64  CSV.open("/home/kent/ruby/SequentialData2-3.csv", "w") do |write|
65          write << ["a", "x0", "x1", "x2"]
66          2.upto(z.size - 2) do |i|
67                  x = main.estimatex(i)
68                  write << [(i + 1) - 15, x[0, 0], x[1, 0], x[2, 0]]
69          end
70  end
```

リスト 4: 逐次型最小二乗法により推定するプログラムの出力結果 (-12$\leq a \leq$15)

```
1   a       x0                      x1                      x2
2   -12     -6246.326748522452      -873.3748393355673      -28.748848079893516
3   -11     -511.43661795399385     -30.835185020904646     1.9968269721245164
4   -10     733.7729785851482       162.50429741683485      9.337427525267325
5   -9      911.9994375364354       190.55362594908433      10.413214943085283
6   -8      807.2042935691519       172.8839317265967       9.698987266696431
7   -7      494.22799827127824      117.19949761791484      7.357390988558048
8   -6      453.9008555497013       107.81795348530211      6.894058783725823
9   -5      446.2777105283755       223.12937400365902      15.449434481672009
10  -4      286.43333286489997      158.600243344935        11.601916579041951
11  -3      525.7213430303264       210.42276141122431      14.052159245815906
12  -2      475.3516584146242       162.72080205633802      10.810485628449378
13  -1      186.88613652946498      441.2105621374848       33.14110668220232
14  0       785.1264551886122       377.78276023718223      24.923412219477704
15  1       -780.9190620523907      631.8481304588638       52.96573836950002
16  2       1724.271511147079       -85.47737001095425      -14.984007676635734
17  3       11551.972212222045      -2780.179796252519      -272.72874281663184
18  4       2530.095376972062       -265.57464652850877     -33.11238065923257
19  5       2753.0789470171494      -331.7312150851032      -39.33359855274518
20  6       2041.1569781089402      -140.10975052249142     -20.936723786740853
21  7       2149.1088200528816      -170.7123331679773      -23.83361422415108
22  8       1798.7737767622195      -74.67628291332582      -14.66404750640147
23  9       1735.8826749491936      -58.74305578486385      -13.112706160613211
24  10      1631.8804769442418      -31.479157911499097     -10.48246273125957
25  11      1596.5330255602069      -22.52431753762925      -9.611925412753736
26  12      1536.5591433511056      -7.232764133513408      -8.127842259139648
27  13      1510.2846252668292      -0.6765934226696624     -7.488604462334487
28  14      1471.4203775391989      8.989961116830639       -6.545474340534313
29  15      1451.5588061032943      13.845999556670986      -6.06993312004548
```

リスト 4 より，時刻 a が 15 のとき推測値は $x_0 = -1452, x_1 = 13.85 x_2 = -6.070$　となっており，正しく推定できていないことがわかる．これは，バッチ型最小二乗法を用いて時刻 a が-15 から-13 までに推定した，推定値および推定誤差共分散の初期値が正確でないことが考えられる．そこで，推定する時刻 a を-13 から-12 までに延長し，再度，バッチ型最小二乗法により推定値および推定誤差共分散を求める．その値を用いて，時刻 a が-11 のときから逐次型最小二乗法により未知数を推定する．時刻 a が-11 のときの推定値は $\hat{\boldsymbol{x}}_{(-11)} = \begin{bmatrix} -510.0 \\ -68.60 \\ -1.587 \end{bmatrix}$, $\boldsymbol{P}_{(-11)} = \begin{bmatrix} 1.714 \times 10^5 & 2.459 \times 10^4 & 878.0 \\ 2.459 \times 10^4 & 3528 & 126.0 \\ 878.0 & 126.0 & 4.500 \end{bmatrix}$ である．出力結果をリスト 5 に示し，推定値の時系列グラフを図 3 に示す．

リスト 5: 逐次型最小二乗法により推定するプログラムの出力結果 (-11≤ a ≤15)

```
1   a       x0                x1                x2
2   -11     9.6391447967      5.8410019516      1.0683052968
3   -10     24.0463204805     8.0309432787      1.1505680406
4   -9      -14.0785217098    2.108107961       0.9239658621
5   -8      -20.2107071278    1.1042885851      0.8839005024
6   -7      4.6456466926      5.2838120713      1.0542389347
7   -6      1.3744705844      4.702509535       1.0295475357
8   -5      -3.2507151087     3.8521056325      0.992533514
9   -4      -4.9882801704     3.51192045        0.9770789431
10  -3      -3.7681279257     3.7614168869      0.9887361849
11  -2      -3.147169859      3.8982165235      0.9954236163
12  -1      -2.2478158233     4.1078165486      1.0060012965
13  0       -2.1925478658     4.1219464603      1.0067496508
14  1       -2.5858819468     4.0135105348      1.0007970835
15  2       -2.2609584533     4.1147281132      1.0066501073
16  3       -2.5596808291     4.0111017006      1.0004115281
17  4       -2.5216806836     4.0268399856      1.0014141957
18  5       -2.6949792434     3.9415369403      0.9957272848
19  6       -2.7199463704     3.924818206       0.994541054
20  7       -2.6863353061     3.9572590087      0.9969644461
21  8       -2.6822132019     3.9675895562      0.9977916824
22  9       -2.6768014729     3.934348456       0.9949678792
23  10      -2.6687488221     3.9258258613      0.9941845421
24  11      -2.7307262247     3.9631202664      0.9978565638
25  12      -2.7365515487     3.965303717       0.9980920932
26  13      -2.8413151212     3.9941550888      1.0014701165
27  14      -2.8336469908     3.9926242626      1.0012702759
28  15      -2.7734278289     3.9832142688      0.9999115648
```

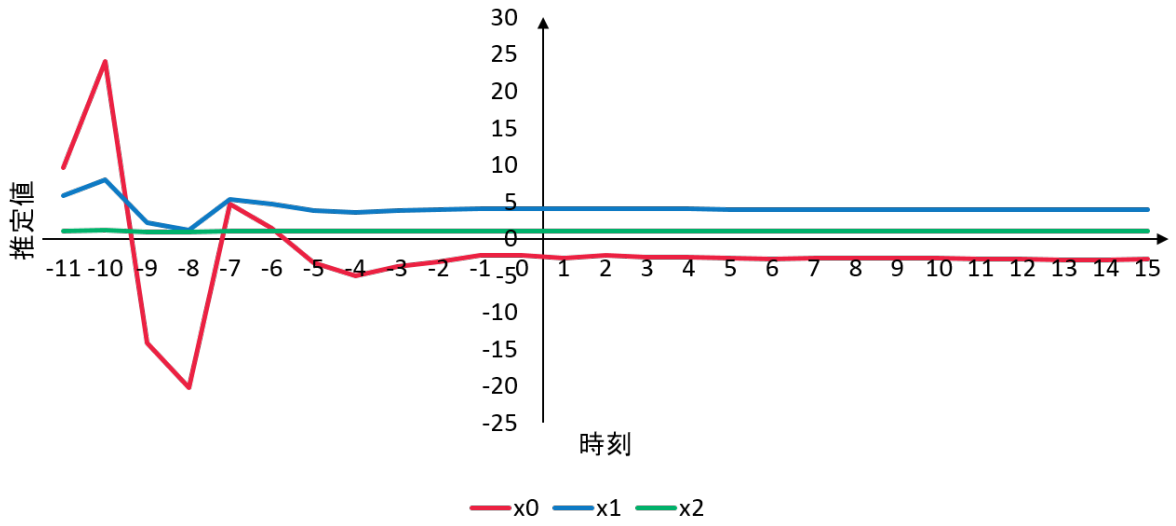

図 3: 逐次型最小二乗法による推定値の時系列グラフ (-11≤ a ≤15)

リスト 5 より，時刻 a が 15 のとき推定値は $x_0 = -2.773, x_1 = 3.983 x_2 = 0.9999$　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 3 より，推定値が収束していることがわかる．

3　逐次型最小二乗法における初期推定誤差共分散に対する推定精度

逐次型最小二乗法では，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，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．このとき推定誤差は，$\boldsymbol{P}_{(0)} = \begin{bmatrix} p & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p \end{bmatrix}$ となるように設定する．$p = 10^{-6}, 10^{-3}, 10^{0}, 10^{3}, 10^{6}$ であるときの，逐次型最小二乗法による推定値の時系列グラフを図 4-8 に示す．

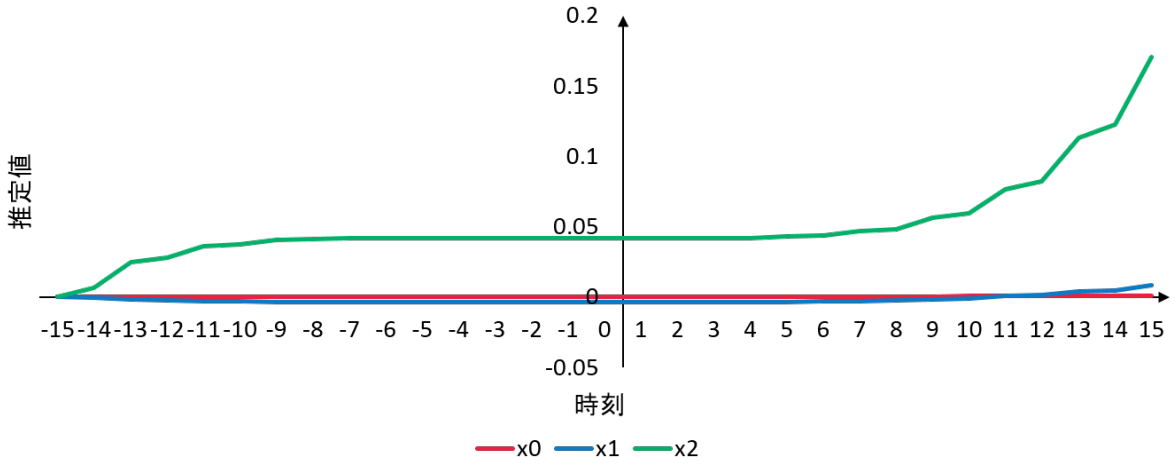

図 4: 逐次型最小二乗法による推定値の時系列グラフ $(p = 10^{-6})$

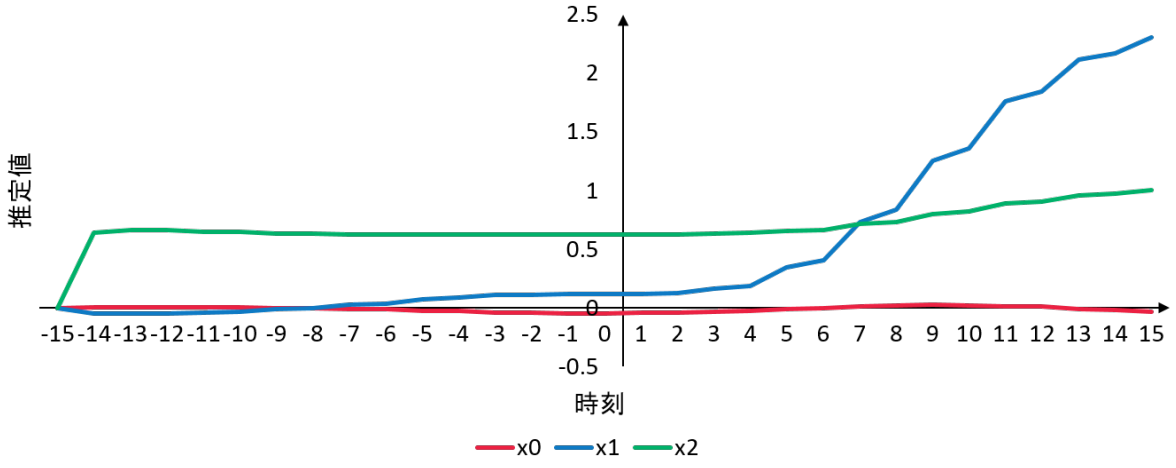

図 5: 逐次型最小二乗法による推定値の時系列グラフ $(p = 10^{-3})$

図 4-8 より，初期推定誤差共分散の値が小さくなればなるほど推定する時間が長くなるため，収束することができないので推定できないことがわかる．一方，初期推定誤差共分散の値が大きくなれば大きくなるほど，最初の時刻での誤差が非常に大きくなるが，その後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．

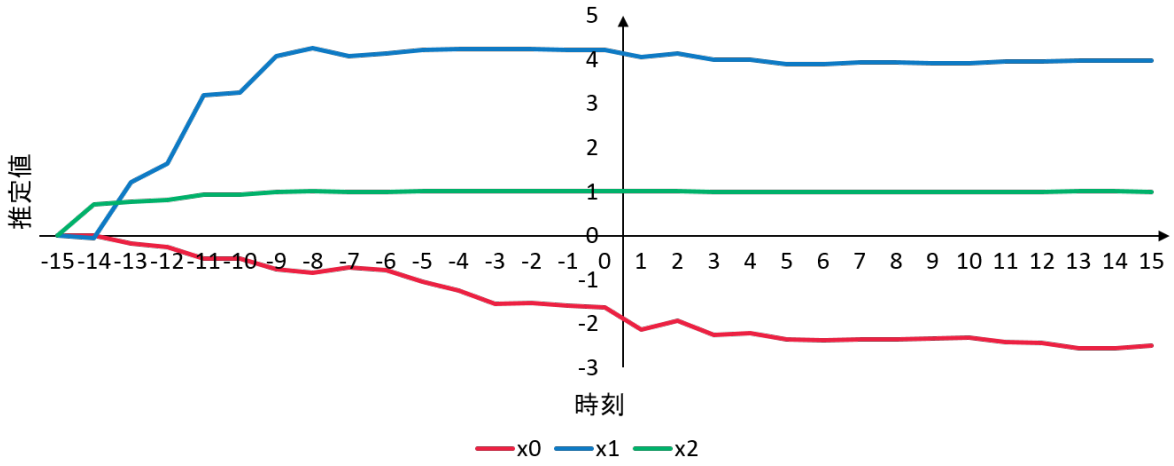

図 6: 逐次型最小二乗法による推定値の時系列グラフ ($p = 10^0$)

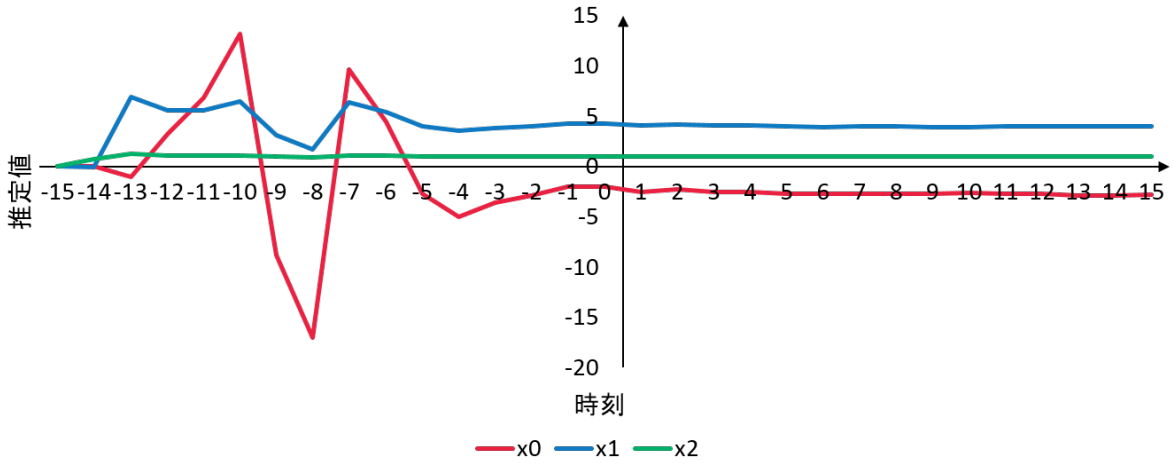

図 7: 逐次型最小二乗法による推定値の時系列グラフ ($p = 10^3$)

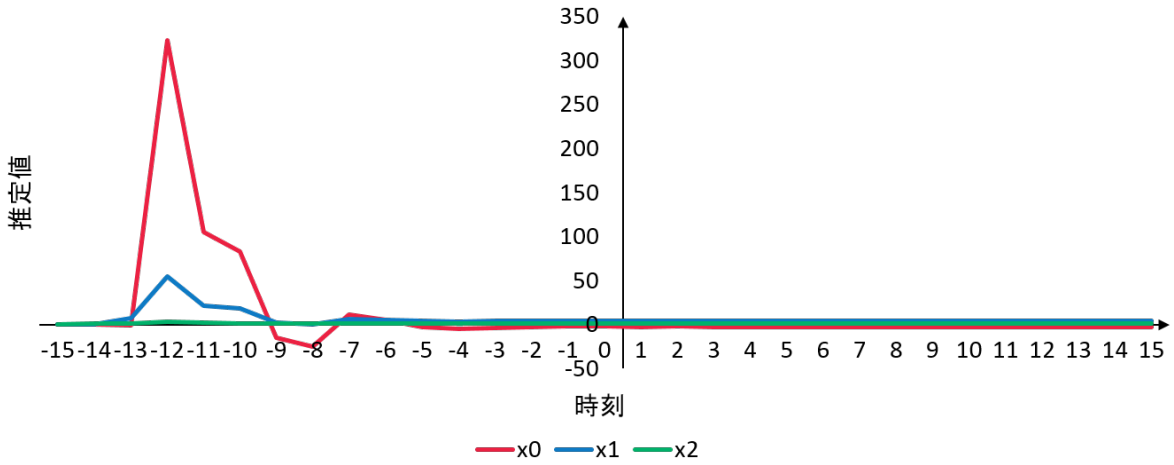

図 8: 逐次型最小二乗法による推定値の時系列グラフ ($p = 10^6$)