確率システム制御特論
課題 2

16344216
田中 良道

2016 年 10 月 11 日

問 1

式 (1) を ARMA モデルに変換する．

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} -0.7 & 0 \\ 0 & -0.3 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} v(k)$$
$$y(k) = \begin{bmatrix} -2 & 3 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{1}$$

これを x_1，x_2 のそれぞれについて求めると，

$$\begin{cases} zx_1(k) = -0.7x_1(k) + v(k) \\ zx_2(k) = -0.3x_2(k) + v(k) \end{cases}$$
$$\begin{cases} x_1(k) = (z+0.7)^{-1}\, v(k) \\ x_2(k) = (z+0.3)^{-1}\, v(k) \end{cases} \tag{2}$$

となる．また，式 (1) より，

$$y(k) = -2x_1(k) + 3x_2(k)$$

であるので，

$$\begin{aligned} y(k) &= \frac{-2}{z+0.7}v(k) + \frac{3}{z+0.3}v(k) \\ &= \frac{z+0.15}{(z+0.7)(z+0.3)}v(k) \\ &= \frac{z+0.15}{z^2+z+0.21}v(k) \\ y(k) &= \frac{z^{-1}+0.15z^{-2}}{1+z^{-1}+0.21z^{-2}}v(k) \end{aligned} \tag{3}$$

となり，与式を ARMA モデルに変換することができる．

問 2

式 (4) に示す AR モデルを用いて推定を行う．

$$\hat{y}(k) = -a_1 y(k-1) - a_2 y(k-2) + v(k),\ k = 2, \cdots, N \tag{4}$$

ただし，$y(k)$ は観測値（真値），$\hat{y}(k)$ は推定値，$v(k)$ は白色雑音とする．いま，

$$\theta \equiv [a_1\ a_2]^T,\ \psi(k) \equiv [-y(k-1)\ -y(k-2)]^T \tag{5}$$

とおくと，

$$\hat{y}(k) = \theta^T \psi(k) + v(k) \tag{6}$$

となる．ここで一括処理最小二乗法を適用してパラメータ θ の同定を行う．つまり，

$$J_N \equiv \frac{1}{N} \sum_{k=2}^{N} \left(y(k) - \theta^T \psi(k) \right)^2 \tag{7}$$

とおくとき，J_N が最小となる θ を求める．式 (7) を展開すると，

$$J_N = \frac{1}{N} \sum_{k=2}^{N} y^2(k) - 2\theta^T \left(\frac{1}{N} \sum_{k=2}^{N} y(k)\psi(k) \right) + \theta^T \left(\frac{1}{N} \sum_{k=2}^{N} \psi(k)\psi^T(k) \right) \theta \tag{8}$$

となる．ここで，

$$c_N \equiv \frac{1}{N} \sum_{k=2}^{N} y^2(k),\ h_N \equiv \frac{1}{N} \sum_{k=2}^{N} y(k)\psi(k),\ G_N \equiv \frac{1}{N} \sum_{k=2}^{N} \psi(k)\psi^T(k) \tag{9}$$

とおくと，

$$J_N = c_N - 2\theta^T h_N + \theta^T G_N \theta \tag{10}$$

となる．いま，J_N が最小となる θ を求める必要があるので，式 (10) の両辺を θ で微分した値が零となる θ を求める．

$$\begin{aligned} \frac{dJ_N}{d\theta} &= 0 - 2h_N + 2G_N\theta = 0 \\ \theta &= {G_N}^{-1} h_N\ \text{（ただし，} det(G_N) \neq 0 \text{ とする．）} \end{aligned} \tag{11}$$

以上より，N 個すべてのデータを用いて推定を行う場合には上記を用いれば良い．しかし，実際の観測ではすべてのデータが観測できているとは限らず，逐次的に推定を行うことが一般的である．そこで，逐次処理により変化する $\hat{\theta}$ を新たにおくと，これは式 (11) と同様に，

$$\hat{\theta} = {G_N}^{-1} h_N\ \text{（ただし，} det(G_N) \neq 0 \text{ とする．）} \tag{12}$$

により求められる．

本問では，sin 波と sin 波と cos 波の合成波にそれぞれ白色雑音を加えたデータに対して推定を行った．また，これらを求めるプログラムを C^{++} 言語により作成した．作成したプログラムを付録に示す．

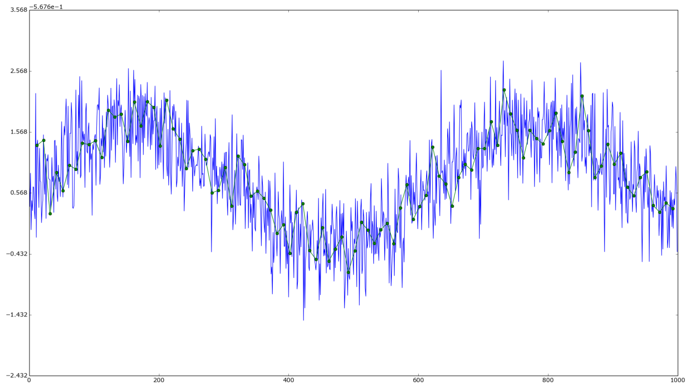

Fig1: 推定結果 1

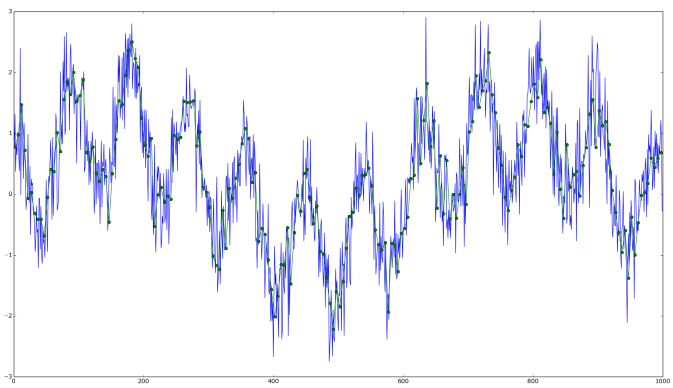

Fig2: 推定結果 2

sin 波

$\sin x + 0.1 + v(k)$ で表されるデータに対して推定を行った．ただし，$v(k)$ は標準偏差 0.5 の正規分布で与えた．Fig. 1 に推定した結果を示す．ただし，青色の実線で示すのが観測値，緑色の点と実線で示すのが推定値である．この時の誤差平均は 0.0707071 であり，分散は 0.188029 であった．

sin 波 + cos 波

$\sin x \cos 5x + 0.1 + v(k)$ で表されるデータに対して推定を行った．ただし，$v(k)$ は標準偏差 0.5 の正規分布で与えた．Fig. 2 に推定した結果を示す．ただし，青色の実線で示すのが観測値，緑色の点と実線で示すのが推定値である．この時の誤差平均は 0.0505051 であり，分散は 0.21554 であった．

付録

Source 1: 作成したプログラム

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <numeric>
#include <Eigen/Dense>

#include "matplotlibcpp.hpp"
namespace plt = matplotlibcpp;

using namespace std;
using namespace Eigen;

void getData(vector<double> &input, string filename)
{
  ifstream ifs(filename.c_str(), ios::in);
  string buf;
  while(1){
        // read by line
        getline(ifs, buf);
        if(ifs.eof())
          break;
        // read by delimiter on reading "one" line
        const char delimiter = ',';
        string spreat_buf;
        istringstream line_separater(buf);
        for(int i=0; ;i++){
          if(line_separater.eof())
                break;
          getline(line_separater, spreat_buf, delimiter);
          input.push_back( atof(spreat_buf.c_str()) );
        }
  }
}

// 一様分布を生成する関数
double sdlab_uniform()
{
  double ret = ((double) rand() + 1.0) / ((double) RAND_MAX + 2.0);
  return ret;
}

// 正規分布を生成する関数
double sdlab_normal(double mu, double sigma)
{
  double z = sqrt(-2.0 * log(sdlab_uniform())) *
    sin(2.0 * M_PI * sdlab_uniform());

  return mu + sigma * z;
}


#define N 1000 // データ点数
```

```
#define RESOLUTION 10 // サンプリング分解能

int main(int argc, char** argv)
{
  vector<double> y_input;  // 観測された y(x)
  vector<double> y_estimate; // 推定値
  vector<double> x; // 用座標 plotx
  vector<double> error; // 推定値誤差

  // === 観測値の代入==========
  // ファイルからデータを読み込む場合
  // getData(y_input, argv[1]);

  // 観測値を関数で与える場合
  for(size_t i=0; i<N; i++){
        // 波 sin
    //y_input.push_back(sin(i*0.01) + 0.1 + sdlab_normal(0, 0.5));
        // sin + cos 波
        y_input.push_back(sin(i*0.01) + cos(i*0.07) + 0.1 + sdlab_normal(0, 0.5));
  }
  // === 観測値代入終了=====

  // Gn, hn, theta_hat, psi の定義と初期化
  // ベクトルはすべて縦ベクトルなので注意
  Matrix2d Gn = Matrix2d::Zero();
  Vector2d hn = Vector2d::Zero();
  Vector2d theta_hat = Vector2d::Zero();
  Vector2d psi = Vector2d::Zero();

  // == 推定ループ開始================================
  for(size_t i=2; i<y_input.size(); i+=RESOLUTION){
        // psi
        psi(0) = -y_input.at(i-1);
        psi(1) = -y_input.at(i-2);
        // Gn
        Gn += psi * psi.transpose();
        // hn
        hn += psi*y_input.at(i);

        Gn /= (double)(i+1);
        hn /= (double)(i+1);

        // 推定パラメータ、推定値の算出
        theta_hat = Gn.inverse() * hn;
        y_estimate.push_back(theta_hat.transpose() * psi);
        vector<double>::iterator itr = y_estimate.end() - 1;
        error.push_back(*itr - y_input.at(i));
        vector<double>::iterator itr2 = error.end() - 1;
        cout << "==================================" << endl;
        cout << "Observed y : " << y_input.at(i) << endl;
        cout << "Estimate y : " << *itr << endl;
        cout << "Error      : " << *itr2 << endl;
        cout << "==================================" << endl;

        *itr2 = fabs(*itr2);
        x.push_back(i);

        Gn *= (double)(i+1);
        hn *= (double)(i+1);
```

```
    }


    // 誤差平均を求める
    double avg_error = accumulate(error.begin()+1, error.end(), 0) / (double)(error.size()-1);
    double cov;
    for(size_t i=1; i<error.size(); i++){
        cov += pow(error.at(i) - avg_error,2);
    }
    cov /= (double)(error.size()-1);
    cout << "Avg Errog : " << avg_error << endl;
    cout << "Cov : " << cov << endl;
    cout << "Sampled data : " << y_estimate.size() << " datas" << endl;


    //plt::named_plot("log(x)", x, z);
    plt::plot(y_input, "-");
    plt::named_plot("$\hat{y}(k)$",x, y_estimate, "-o");
    plt::show();

    return 0;
}
```