Monte Carlo Dynamic Classifier
プログラム実行手順書

Version 1.2.0
2016/03/29

変更履歴

Version	Date	Description
1.2.0	2016-3-29	- Add a sample folder “sample2015” - Allow to select a spline function instead of “spapi” for 1-dimensional state sequences - Allow to use multiple observation sequences for a known state sequence - Constrain particles within the grid range for the particle filter algorithm - Allow to update the observation function for each dimension at each iteration - Implement the MH with Gibbs sampling algorithm
1.1.9	2015-12-22	- Fix LogMvnPdf.m - Fix scripts for plotting in “sample” folder - Add sample scripts: sample_for_Kitagawa.m, sample_for_Lorenz.m, and sample_for_Motion.m
1.1.9-	2015-12-12	- Delete unnecessary white spaces - Fix the log file format - Modify Graphs.m - Translate Japanese comments in the source codes into English - Update copyright (2014 -> 2014-2015) - Fix sample scripts for random number generation:　add “rng('default')” to reproduce the same results - Add sample scripts for plotting - Add unit test scripts
1.1.8	2014-12-22	Bug fixes in Graphs.m in the case that either the dimension of state or observation variables is equal to one
1.1.7	2014-12-14	Change the notations of “PMCMC” and “PMCMC2” into “PMCMC” and “PMCMC2”, respectively, in all codes
1.1.6	2014-12-08	Change the notations of “PSMC” into “PSMC” in all codes
1.1.5	2014-11-25	Add the license information (GPL v2) in all codes
1.1.4	2014-11-10	Bug fix for one-dimensional state variable
1.1.3	2014-06-30	Initial release

目次

1. はじめに

本書は、Monte Carlo Dynamic Classifier ツールの実行手順を説明するものです。

Monte Carlo Dynamic Classifier ツールは、任意の観測データ系列のモデル推定、およびその推定モデルによる状態系列の推定を行うプログラムです。推定されたモデルは、異なる観測データ系列に適用してモデルの尤度を計算することにより、観測データ系列のクラス分類などに応用できます。MCDC ツールは、以下のプログラム群から構成されます。

MCDCTrain
: モデル推定プログラム

MCDCTest
: モデルの尤度計算プログラム

Graphs
: 推定されたモデルのグラフ描画関数群

本書の以下の各章では、これらのプログラムの実行方法と実行例（第 2 章)、プログラム構成（第 3 章）について説明します。また、本ツールに付属するサンプルプログラムの一部では、CMU Graphics Lab Motion Capture Database にて公開されているモーションキャプチャデータを利用します。このデータの利用許諾条件について第 4 章に記載します。

2. プログラムの実行

MCDC ツールでは、観測データ系列に対してモデル推定を行う MCDCTrain と、推定モデルを用いて未知の観測データ系列の状態系列の推定を行う MCDCTest が提供されます。

このほかに、推定モデルをグラフ描画する関数群を集めたツールとして Graphs クラスが提供されます。本章ではこれらのプログラムの実行方法について説明します。

2.1. MCDCTrain

MCDCTrain 関数は、観測データ系列に対してモデル推定を行うプログラムです。

2.1.1. 実行方法

MCDCTrain 関数は、以下のように実行します。

```
[ IDX, SKP, OKP, FV, GV, XE, YE, loglik ] = MCDCTrain( ...
  algorithm, ...
  grids, ...
  stateKernelGens, ...
  obsKernelGens, ...
  stateMeanFuncs, ...
  obsMeanFuncs, ...
  gridDimForGramMatrix, ...
  stateSplineHandle, ...
  obsSplineHandle, ...
  x0, ...
  xaux, ...
  u, ...
  y', ...
  N, ...
  J, ...
  K, ...
  aspect ...
);
```

また、以前の実行で出力されたファイルを読み込み、反復実行を継続することもできます。この場合は、MCDCTrain 関数を以下のように実行します。継続実行の際には、指定された matfile からパラメータを読み込み、前回と同じ設定で実行を再開します。ただし、Name, Value の組を指定することで、前回の設定を上書きして実行できます。

```
[IDX, SKP, OKP, FV, GV, XE, YE, loglik] = MCDCTrain(matfile, aspect, Name, Value. ...)
```

2.1.2. パラメータ

MCDCTrain 関数のパラメータは以下のとおりです。

パラメータ	データ型	内容
algorithm	Algorithm	アルゴリズム種別
grids	G double[] cell	状態空間上に構成される格子点の座標。各次元の値を double 配列として格納したセル配列として表す
stateKernelGens	Dx handle cell	状態遷移関数のカーネル事前分布。Dx は状態空間のうち推定対象とされる次元数とする
obsKernelGens	P handle cell	観測関数のカーネル事前分布
gridDimForGramMatrix	int	グラム行列作成に用いる次元
stateSplineHandle	handle	状態遷移関数のスプライン補間に用いるアルゴリズム
obsSplineHandle	handle	観測関数のスプライン補間に用いるアルゴリズム
x0	Dx*1 double	状態変数の初期状態
xaux	Da*T double	追加状態データ
u	D*T double	制御データ
z	P*T double	観測データ
N	int	Particle filter 実行時の粒子数
J	int	MCMC iteration の全体の反復回数
K	int	観測オフセット
aspect	Aspect	システム設定情報（ログ出力先等）

algorithm パラメータには、以下のいずれかのクラスを選択できます。

クラス名	アルゴリズム内容
PSMC	状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定は行わない
PMCMC	状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定に Particle marginal Metropolis-Hastings 法を用いる
PMCMC2	状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定に Particle marginal Metropolis-Hastings 法を用いる。状態空間の特定の一次元のみを共分散関数に用いる

アルゴリズムとして PMCMC2 を利用する場合には、さらに、平均値関数と共分散関数の学習方式、共分散関数に用いるカーネルパラメータの学習方式を以下から選択できます。

クラス名	アルゴリズム内容
MCDCStrategyChoice1	平均値関数を学習せず、常にアンカーモデルを利

クラス名	アルゴリズム内容
	用する。共分散関数にカーネル関数を用いる
MCDCStrategyChoice2	現在の GP surface を平均値関数として関数をサンプリングする。共分散関数にカーネル関数を用いる
MCDCStrategyChoice3	現在の GP surface を平均値関数として関数をサンプリングする。共分散関数にカーネル関数を用いず、固定された共分散行列を用いる
RBFKernelGeneratorStrategyChoice1	RBF カーネルのカーネルパラメータを事前分布からランダムサンプリングする
RBFKernelGeneratorStrategyChoice2	RBF カーネルのカーネルパラメータを現在値からのランダムウォークによって生成する

stateSplineHandle, obsSplineHandle パラメータには、自ら設計したスプライン関数のハンドルを渡せますが、既定のスプライン関数として以下を用意しています。

関数名	アルゴリズム内容
GenericSpline	多次元の潜在状態空間で利用できる一般的なスプライン補間の実装。内部で spapi 関数を利用する
SimpleSpline	一次元の潜在状態空間で利用できるスプライン補間の実装。GenericSpline よりも高速に動作する。内部では spline 関数を利用する
SimpleSplineInGrid	一次元の潜在状態空間における状態遷移関数に利用できるスプライン補間の実装。遷移先がグリッドの範囲外に出てしまったときに、グリッドの端点に引き戻す処理を行う

2.1.3. 戻り値

MCDCTrain 関数の実行結果として戻される値は、以下のとおりです。

値	データ型	内容
IDX	J*1 double	j 回目の反復までの累積受理回数
SKP	A*D*2 double	第 a 受理反復[1]における、状態遷移関数のカーネルパラメータ（sigma, l）
OKP	A*P*2 double	第 a 受理反復における、観測関数のカーネルパラメータ（sigma, l）
FV	A*D*G double	第 a 受理反復における、状態遷移関数による格子点上の値。G は格子点のサイズに一致する多次元配列

[1] MCMC iteration において、第 a 回目に受理されたときの反復をこのように表記します。

値	データ型	内容
GV	A*P*G double	第 a 受理反復における、観測関数による格子点上の値。G は格子点のサイズに一致する多次元配列
XE	A*T*D double	第 a 受理反復における、時刻 t の状態変数の推定平均
YE	A*T*P double	第 a 受理反復における、時刻 t の観測変数の推定平均
loglik	A*1 double	第 a 受理反復における、particle filter による推定の対数尤度

2.1.4. 中間ファイル

実行時には、aspect の設定にしたがって中間状態が保存されます。これは.mat 形式のダンプファイルとして出力されます。ファイルに格納されているデータは以下のとおりです。

値	データ型	内容
algorithm	Algorithm	MCDCTrain のパラメータに渡された algorithm
in	MCDCInput	MCDCTrain のパラメータのうち、algorithm と aspect を除くすべて
out	MCDCOutput	MCDCTrain の戻り値のすべて（途中状態）および、現在の反復回数 j

2.1.5. ログファイル

実行中には、aspect の設定にしたがってログファイルが出力されます。これはテキスト形式のファイルになります。ログファイルは以下のような形式になります。

```
2014/04/26 11:29:36 - Iteration 18 / 200
2014/04/26 11:29:36 -   StateKernel[1]: [ Sigma=8.932992, L=7.409991 ]
2014/04/26 11:29:36 -   StateKernel[2]: [ Sigma=4.410584, L=2.304895 ]
2014/04/26 11:29:36 -   ObsKernel[1]: [ Sigma=1.711191, L=9.248074 ]
2014/04/26 11:29:36 -   ObsKernel[2]: [ Sigma=9.867704, L=9.599577 ]
2014/04/26 11:29:36 -   ObsKernel[3]: [ Sigma=7.700470, L=8.679293 ]
2014/04/26 11:29:36 -   Creating GramMatrix using 1 dim...
2014/04/26 11:29:36 -     StateKernel[1] Done
2014/04/26 11:29:36 -     StateKernel[2] Done
2014/04/26 11:29:36 -     ObsKernel[1] Done
2014/04/26 11:29:36 -     ObsKernel[2] Done
2014/04/26 11:29:36 -     ObsKernel[3] Done
2014/04/26 11:29:36 -   Drawing GP surface...
2014/04/26 11:29:36 -   Estimating using particle filter... (N=500)
2014/04/26 11:30:20 -   Acceptance log probability = 232745.510426
2014/04/26 11:30:20 -   logLH = -5347068.232758, accepted
2014/04/26 11:30:20 - Elapsed time is 44.324939 seconds.
```

```
2014/04/26 11:30:20 -   j: 8 bytes
2014/04/26 11:30:20 -   IDX: 1600 bytes
2014/04/26 11:30:20 -   SKP: 128 bytes
2014/04/26 11:30:20 -   OKP: 192 bytes
2014/04/26 11:30:20 -   FV: 61504 bytes
2014/04/26 11:30:20 -   GV: 92256 bytes
2014/04/26 11:30:20 -   XE: 122752 bytes
2014/04/26 11:30:20 -   YE: 184128 bytes
2014/04/26 11:30:20 -   loglik: 32 bytes
```

2.2. MCDCTest

MCDCTest 関数は、MCDCTrain によって得られたモデルを用いて未知のデータの状態を推定します。複数の異なるモデルを用いて状態推定を行い、それぞれの尤度を比較することで、クラス分類問題への応用も可能です。

2.2.1. 実行方法

MCDCTest 関数は、以下のように実行します。

```
[ result, FnState, FnObs ] = MCDCTest(u, y, N, modelFile)
```

2.2.2. パラメータ

MCDCTest 関数のパラメータは以下のとおりです。

パラメータ	データ型	内容
u	D*T double	制御データ
y	P*T double	観測データ
N	int	Particle filter 実行時の粒子数
modelFile	chars	モデルファイル

2.2.3. 戻り値

MCDCTest 関数の実行結果として戻される値は、以下のとおりです。

値	データ型	内容
result	ParticleFilter	j 回目の反復までの累積受理回数
FnState	handle	状態遷移関数
FnObs	handle	観測関数

戻り値の result に、particle filter による状態推定の結果が含まれます。これは以下の構造を持ちます。

値	データ型	内容
Particles	N*T*D double	時刻 t における各粒子の座標
Weights	N*T double	時刻 t における各粒子の重み
Loglik	double	推定された状態の対数尤度

2.3. Graphs

MCDCTrain によって推定されたモデルは、Graph クラスに定義された関数群を用いて PDF ファイルに出力することができます。以下のグラフを出力可能です。

2.3.1. Graphs.YE

観測データ系列と、推定モデルを用いた粒子フィルタによって観測データを追跡した結果の時間推移グラフを出力します。MCMC iteration の中で、指定された特定の反復における推定モデルを用いた結果を出力します。観測データ系列の次元ごとにグラフが出力されます。以下のように実行します。

```
Graphs.YE( ...
  outputFileNamePrefix, ...
  matFileName, ...
  iterations, ...
  times ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。ファイル名は、指定された接頭辞に、次元数と拡張子.pdf を付け加えたものになります
matFileName	char[]	推定モデルを含む Matlab データファイル名
iterations	int[]	プロットする推定値の MCMC iteration を列挙した配列
times	int[]	プロットするデータ系列の範囲。省略時はデータ系列の全体を出力します

2.3.2. Graphs.XE

推定モデルを用いた粒子フィルタによる状態系列推定値の時間推移グラフを出力します。MCMC iteration の中で、指定された特定の反復における推定モデルを用いた結果を出力します。状態系列の次元ごとにグラフが出力されます。以下のように実行します。

```
Graphs.XE( ...
  outputFileNamePrefix, ...
  matFileName, ...
  iterations, ...
  times ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。フ

パラメータ	データ型	内容
		ァイル名は、指定された接頭辞に、次元数と拡張子.pdf を付け加えたものになります
matFileName	char[]	推定モデルを含む Matlab データファイル名
iterations	int[]	プロットする推定値の MCMC iteration を列挙した配列
times	int[]	プロットするデータ系列の範囲。省略時はデータ系列の全体を出力します

2.3.3. Graphs.YEMean

観測データ系列と、推定モデルを用いた粒子フィルタによって観測データを追跡した結果の時間推移グラフを出力します。MCMC iteration の全体を平均した推定モデルを用います。観測データ系列の次元ごとにグラフが出力されます。以下のように実行します。

```
Graphs.YEMean( ...
  outputFileNamePrefix, ...
  matFileName, ...
  times ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。ファイル名は、指定された接頭辞に、次元数と拡張子.pdf を付け加えたものになります
matFileName	char[]	推定モデルを含む Matlab データファイル名
times	int[]	プロットするデータ系列の範囲。省略時はデータ系列の全体を出力します

2.3.4. Graphs.XEMean

推定モデルを用いた粒子フィルタによる状態系列推定値の時間推移グラフを出力します。MCMC iteration の全体を平均した推定モデルを用います。状態系列の次元ごとにグラフが出力されます。以下のように実行します。

```
Graphs.XEMean( ...
  outputFileNamePrefix, ...
  matFileName, ...
  times ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。フ

パラメータ	データ型	内容
		ファイル名は、指定された接頭辞に、次元数と拡張子.pdf を付け加えたものになります
matFileName	char[]	推定モデルを含む Matlab データファイル名
times	int[]	プロットするデータ系列の範囲。省略時はデータ系列の全体を出力します

2.3.5. Graphs.Loglik

モデル推定において、MCMC iteration ごとに推定されたモデルの対数尤度を出力します。以下のように実行します。

```
loglik = Graphs.Loglik( ...
  outputFileNamePrefix, ...
  matFileName1, ...
  matFileName2, ...
  ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。ファイル名は、指定された接頭辞に.pdf を付け加えたものになります
matFileName1, 2, ...	char[]	推定モデルを含む Matlab データファイル名。複数ファイルを指定した場合は、それぞれを一系列としてグラフを描画します。

関数の戻り値は以下のとおりです。

戻り値	データ型	内容
loglik	double	モデルの対数尤度です

2.3.6. Graphs.Rmse

モデル推定において、各 MCMC iteration までの推定モデルを用いて観測データ系列を推定した結果の平均二乗誤差を出力します。以下のように実行します。

```
[RmseMean, RmseStd, Rmse] = Graphs.Rmse( ...
  outputFileNamePrefix, ...
  matFileName ...
);
```

パラメータは以下のとおりです。

パラメータ	データ型	内容
outputFileNamePrefix	char[]	出力グラフファイルのファイル名接頭辞。ファイル名は、指定された接頭辞に.pdf を付け加えたものになります
matFileName	char[]	推定モデルを含む Matlab データファイル名

関数の戻り値は以下のとおりです。

戻り値	データ型	内容
RmseMean	double	各 MCMC iteration までのモデルで推定した平均二乗誤差の平均です
RmseStd	double	各 MCMC iteration までのモデルで推定した平均二乗誤差の標準偏差です
Rmse	double[]	各 MCMC iteration までのモデルで推定した場合の平均二乗誤差の行列です

2.4. プログラム実行例

MCDCTrain によるモデル推定の実行例として、Kitagawa モデルから生成した観測データ系列の推定を行う例を示します。本節で説明する内容による実行例は以下にあります。

```
samples/KitagawaModelPMCMC2Estimation.m
```

2.4.1. 観測データの生成

まず、モデル推定の対象とする観測データを用意します。本来、観測データは事前に与えられるものですが、ここでは Kitagawa モデルから生成したデータ系列を観測データとして利用します。

```
[x, y] = KitagawaModel(1000, 0.5, 28, 8, 0.6, 30, 10, 0.05, 0.06, 0.07, 0.08, 0.1, 0.1);
u  = repmat(cos(1.2 * [1:T]), 2, 1)';
```

このコードにより、y に観測データ系列が格納されます。x には状態の系列が格納されますが、x はこの後の処理では利用しません。また、Kitagawa モデルでは、時変の制御データを与えるため、観測データの生成に合わせて、制御データ系列もここで生成しています。

2.4.2. 状態空間のデザイン

MCDCTrain は、状態空間モデルを未知としてモデル推定を行います。現在のプログラムでは、状態空間の次元数、状態変数の値が動く範囲を与える必要があります。以下では、二次元の状態空間を考え、各次元について-30 から 30 の範囲で 2.0 刻みの格子点を設定します。

```
grids = { ...
  [-30:2:30], ...
  [-30:2:30]  ...
```

```
};
```

モデル推定の処理では、ここで設定された格子点上でガウス過程から関数の値をサンプリングし、それをスプライン補間することで状態遷移関数、観測関数を作ります。格子点をより多く、より広範に取ることでモデルの表現力は高くなりますが、処理時間、メモリ使用量が大幅に増えることになります。

2.4.3. モデルのデザイン

次に、状態遷移関数、観測関数のモデルをデザインします。MCDCTrain では、状態遷移関数と観測関数のそれぞれについて、ガウス過程から関数をサンプリングする際の平均値関数、カーネル共分散行列を与えることができます。

```
stateKernelGens = { ...
  RBFKernelGenerator(UniformDistribution(0.01, 10), UniformDistribution(0.01, 10)), ...
  RBFKernelGenerator(UniformDistribution(0.01, 10), UniformDistribution(0.01, 10))  ...
};

obsKernelGens = {
  RBFKernelGenerator(UniformDistribution(0.01, 10), UniformDistribution(0.01, 10)), ...
  RBFKernelGenerator(UniformDistribution(0.01, 10), UniformDistribution(0.01, 10)), ...
  RBFKernelGenerator(UniformDistribution(0.01, 10), UniformDistribution(0.01, 10))  ...
};

stateMeanFuncs = { ...
  @(x) (a1 .* x(1,:) + b1 .* x(1,:) / (1 + x(1,:) .^ 2)), ...
  @(x) (a2 .* x(2,:) + b2 .* x(2,:) / (1 + x(2,:) .^ 2))  ...
};

obsMeanFuncs = { ...
  @(x) (d1 .* x(1,:) .^ 2 + d2 .* x(2,:) .^ 2), ...
  @(x) (d3 .* x(1,:) .^ 2                    ), ...
  @(x) (                   d4 .* x(2,:) .^ 2)  ...
};
```

上記のコード例では、パラメータσ、lがともに [0.01,10]の一様分布に従う RBF カーネルを用い、平均値関数には Kitagawa モデルの状態遷移関数、観測関数を設定しています[2]。

2.4.4. MCDCTrain のアルゴリズム設定

MCDCTrain の MCMC iteration で利用するアルゴリズムを選択します。カーネルパラメータ、平均値関数を事前分布に固定するか、毎回遷移させるかを選択できます。

```
kernelGeneratorStrategy = @RBFKernelGeneratorStrategyChoice2;
mcdcStrategy = @MCDCStrategyChoice2;
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);
```

[2]一般的には、真のモデルが未知であるため、平均値関数として真のモデルを設定することはできません。

反復計算でギブスサンプリングを用いず、通常の MCMC を実行することを指定します。ギブスサンプリングを併用する方法の詳細は 2.5.6 節を参照してください

```
algorithm.SetPMCMCProbability(1.0);
```

状態遷移関数、観測関数を未知としてガウス過程による推定を行うため、次のように設定します。いずれかの関数が既知である場合は、推定を行わずに実行することも可能です。詳細は 2.5.1 節、2.5.2 節を参照してください。

```
stateModel = GaussianProcessModel(ModelKind.State, algorithm);
algorithm.SetStateModel(stateModel);
obsModel = GaussianProcessModel(ModelKind.Observation, algorithm);
algorithm.SetObsModel(obsModel);
```

その他の設定です。MCDCStrategyChoice2 を選択した場合は、潜在状態空間の中で独立でないとみなす次元を一つ選択します。また、ここで利用するデータは多次元の潜在状態空間を持つので、スプライン補間は GenericSpline を指定します。

```
gridDimForGramMatrix = 1;
stateSplineHandle = @GenericSpline;
obsSplineHandle = @GenericSpline;
```

2.4.5. モデル推定の実行に関する設定

モデル推定の反復回数、粒子数、また、ログファイルの出力先等の設定を行います。

```
x0 = zeros(1, 2);
N  = 500;
J  = 100000;

aspect = Aspect();
aspect.LogFileName = 'logs/KitagawaModelPMCMC2.log';
aspect.MatFileNamePrefix = 'logs/KitagawaModelPMCMC2';
aspect.SavesIntermediateMat = true;
aspect.IntermediateMatInterval = 100;
```

潜在状態系列を未知として粒子フィルタによる推定を行うため、次のように設定します。系列が既知の場合や、複数のデータ系列を利用できる場合のプログラム実行方法については、2.5.3 節を参照してください。

```
likelihoodCalculator = PMCMCParticleFilter(x0, xaux, u, y, q, r, N, 0);
algorithm.SetLikelihoodCalculator(likelihoodCalculator);
```

2.4.6. モデル推定の実行

ここまでの設定を用いて、MCDCTrain によるモデル推定を実行します。

```
[ IDX, SKP, OKP, FV, GV, XE, YE, loglik ] = MCDCTrain( ...
  algorithm, …
  grids, ...
```

```
  stateKernelGens, ...
  obsKernelGens, ...
  stateMeanFuncs, ...
  obsMeanFuncs, ...
  gridDimForGramMatrix, ...
  stateSplineHandle, ...
  obsSplineHandle, ...
  x0, ...
  [], ... % No auxiliary states
  u', ...
  y', ...
  N, ...
  J, ...
  0, ...
  aspect ...
);
```

2.5. MCDCTrain の高度な利用方法

MCDCTrain プログラムを実行する際、Algorithm オブジェクトを適切に設定することにより、既定の動作と異なる処理を行わせることができます。本節では、そのような高度な実行方法について説明します。

2.5.1. 状態遷移モデルを外部から設定する

MCDCTrain で推定する状態空間モデルにおいて、状態遷移モデルが既知の場合には、外部からモデルを与えることができます。モデルが与えられた場合には、MCDCTrain では状態遷移モデルの推定を行わず、観測モデルの推定のみを行います。

外部から状態遷移モデルを与えるには、Algorithm クラスの SetStateModel メソッドを利用します。コード例を以下に示します。

```
% 状態遷移モデルを無名関数の配列として定義する
stateMeanFuncs = { ...
  @(x) (a1 .* x(1,:) + b1 .* x(1,:) / (1 + x(1,:) .^ 2)), ...
  @(x) (a2 .* x(2,:) + b2 .* x(2,:) / (1 + x(2,:) .^ 2))  ...
};

% Algorithm オブジェクトを生成する
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Algorithm オブジェクトに状態遷移モデルを設定する
stateModel = FixedModel(ModelKind.State, VectorValuedFunction(stateMeanFuncs));
algorithm.SetStateModel(stateModel);
```

ここで説明した内容による実行例は、以下にあります。

```
samples/KitagawaModelPMCMCEstimation_stateFixed.m
```

2.5.2. 観測モデルを外部から設定する

MCDCTrain で推定する状態空間モデルにおいて、観測モデルが既知の場合には、外部からモデルを与えることができます。モデルが与えられた場合には、MCDCTrain では観測モデルの推定を行わず、状態遷移モデルの推定のみを行います。

外部から観測モデルを与えるには、Algorithm クラスの SetObsModel メソッドを利用します。コード例を以下に示します。

```
% 観測モデルを無名関数の配列として定義する
obsMeanFuncs = { ...
  @(x) (d1 .* x(1,:) .^ 2 + d2 .* x(2,:) .^ 2), ...
  @(x) (d3 .* x(1,:) .^ 2                    ), ...
  @(x) (                   d4 .* x(2,:) .^ 2)  ...
};

% Algorithm オブジェクトを生成する
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Algorithm オブジェクトに観測モデルを設定する
obsModel = FixedModel(ModelKind.Observation, VectorValuedFunction(obsMeanFuncs));
algorithm.SetObsModel(obsModel);
```

ここで説明した内容による実行例は、以下にあります。

```
samples/KitagawaModelPMCMCEstimation_obsFixed.m
```

2.5.3. 潜在状態系列を外部から設定する

MCDCTrain でのモデル推定では、通常は、観測データの系列のみを与え、粒子フィルタを用いて潜在状態の系列を推定します。ただし、潜在状態の系列も既知の場合には、観測データに合わせて潜在状態データも与えることで、粒子フィルタを利用せずに、与えられた系列の生起確率を計算してモデルの受理確率を計算することができます。潜在状態系列を設定するコード例を以下に示します。

```
% Algorithm オブジェクトを生成する
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% 潜在状態系列を既知として Algorithm オブジェクトに設定する
likelihoodCalculator = KnownSequence(x, xaux, u, y, q, r, K);
algorithm.SetLikelihoodCalculator(likelihoodCalculator);
```

KnownSequence クラスコンストラクタのパラメータは以下のとおりです。

パラメータ	データ型	内容
x	Dx*T double	潜在状態データ
xaux	Da*T double	追加状態データ
u	D*T double	制御データ
y	P*T double	観測データ

パラメータ	データ型	内容
q	double	状態遷移ノイズの分散
r	double	観測ノイズの分散
K	int	観測オフセット

ここで説明した内容による実行例は、以下にあります。

```
samples/KitagawaModelPMCMCEstimation_knownSequence.m
```

2.5.4. 非ガウスノイズを用いて状態系列を推定する

MCDCTrain でのモデル推定では、通常は、システムに加えられるノイズはガウス分布にしたがうものと仮定します。ただし、ガウス分布のほかにもいくつかの確率分布が用意されており、確率分布を選択して Algorithm オブジェクトに設定することで、非ガウスノイズを用いて状態系列を推定することができます。潜在状態系列に加えられるノイズを設定するコード例を以下に示します。

```
% Algorithmオブジェクトを生成する
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% 潜在状態系列に加えられるノイズの確率分布を生成する
xdim = size(x0, 2);
stateNoise = CauchyDistribution(0, q, xdim);

% ノイズの確率分布をAlgorithmオブジェクトに設定する
algorithm.SetStateNoise(stateNoise);
```

観測系列に加えられるノイズを設定するには、同様にノイズの確率分布を生成したのち、Algorithm オブジェクトの SetObsNoise メソッドを用いて確率分布を設定します。

指定可能な確率分布と各々のパラメータは以下のとおりです。末尾のパラメータの dim には、潜在状態または観測の次元数を指定します。

確率分布・パラメータ	内容
CauchyDistribution(m,v,dim)	コーシー分布 (平均 m, 分散 v)
ExponentialDistribution(mu,dim)	指数分布 (平均 mu)
GammaDistribution(a,b,dim)	ガンマ分布 (形状 a, スケール b)
GeneralizedParetoDistribution(k,sigma,theta,dim)	一般化パレート分布 (形状 k, スケール sigma, 位置 theta)
LaplaceDistribution(location,scale,dim)	ラプラス分布 (位置 location, スケール scale)

確率分布・パラメータ	内容
NormalDistribution(m,v,dim)	正規分布 (平均 m, 分散 v)
TDistribution(nu,dim)	t 分布 (自由度 nu)
TLocationScaleDistribution(m,v,nu,dim)	t 位置スケール分布 (平均 m, 分散 v, 自由度 nu)
UniformDistribution(lb,ub,dim)	連続一様分布 (下限 lb, 上限 ub)
WeibullDistribution(a,b,dim)	ワイブル分布 (スケール a, 形状 b)

ここで説明した内容による実行例は、以下にあります。

```
samples/KitagawaModelPMCMCEstimation_nonGaussianNoise.m
```

2.5.5. 複数の既知のデータ系列を外部から設定する

バージョン 1.2 以降の MCDC ツールでは、2.5.3 節の内容を拡張して、複数の既知のデータ系列を外部から与えてモデルパラメータを推定できるようになりました。この設定のコード例を以下に示します。

```
% Algorithm オブジェクトを生成する
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% 複数の既知のデータ系列を Algorithm オブジェクトに設定する
likelihoodCalculator = MultipleKnownSequence(X, xaux, u, Y, q, r, K);
algorithm.SetLikelihoodCalculator(likelihoodCalculator);
```

MultipleKnownSequence クラスコンストラクタのパラメータは以下のとおりです。

パラメータ	データ型	内容
X	Cell of Dx*T double	潜在状態データのセル配列 各セルに Dx*T 要素の行列を格納します
xaux	Cell of Da*T double	追加状態データのセル配列 各セルに Da*T 要素の行列を格納します
u	Cell of D*T double	制御データのセル配列 各セルに D*T 要素の行列を格納します
Y	Cell of P*T double	観測データのセル配列 各セルに P*T 要素の行列を格納します
q	double	状態遷移ノイズの分散
r	double	観測ノイズの分散
K	int	観測オフセット

2.5.6. Metropolis-Hastings with Gibbs sampling による推定

バージョン 1.2 以降の MCDC ツールでは、EM 法の反復計算において、PMCMC 法と MH with Gibbs sampling 法を確率的に選択して実行するようになりました。それぞれの方式が選択される確率を次のように制御できます。0 から 1 の間の値を指定し、0 を指定すると常に MH with Gibbs sampling 法、1 を指定すると常に PMCMC 法になります。未指定の場合の既定値は 0.25 です。

```
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);
algorithm.SetPMCMCProbability(1.0); % Don't use MH-Gibbs
```

3. プログラム構成

本章では、MCDC ツールのプログラム構成について説明します。

3.1. ファイル構成

MCDC ツールは、Matlab のプログラムとして作成されています。プログラムファイルの一覧は以下のとおりです。

ファイル名	内容
Algorithm.m	モデル推定に用いるアルゴリズムを表す抽象基底クラス
Aspect.m	プログラム動作を定める設定クラス
BoundedNormalDistribution.m	正の範囲に限定された正規分布
CauchyDistribution.m	コーシー分布
CheckGridTransformation.m	グリッドの写像に対する範囲チェック関数
CompositeLikelihoodCalculator.m	複数系列の尤度を求めるクラス
ContinuousUnivariateDistribution.m	連続単変量確率分布を表す抽象既定クラス
DBA.m	DTW Barycenter Averaging の実装
DBAAlign.m	DBA を用いて系列長を揃える補助ツール
Distribution.m	確率分布を表す抽象基底クラス
ExponentialDistribution.m	指数分布
FixedModel.m	固定されたモデル。推定を行わない
FixedValueDistribution.m	特定の一点を取る分布
GPSurface.m	グリッドの写像を求める関数
GammaDistribution.m	ガンマ分布
GaussianProcessModel.m	状態空間モデル。ガウス過程による推定を行う
GaussianProcessModelRotate.m	状態空間モデル。ガウス過程による推定を行う。EM アルゴリズムの各反復で関数の一次元のみをサンプリングする
GeneralizedParetoDistribution.m	一般化パレート分布
GenericSpline.m	多次元スプライン補間関数。spapi 関数を用いる
Graphs.m	グラフ描画関数群
GridData.m	状態空間におけるグリッド構造
IterationStrategy.m	EM アルゴリズムの反復方法 (抽象基底クラス)
IterationStrategyDefault.m	PMCMC 法による反復を行う
IterationStrategyMHGibbs.m	MH with Gibbs sampling による反復を行

ファイル名	内容
	う
IterationStrategyProbabilistic.m	PMCMC 法と MH with Gibbs sampling 法を確率的に選択する
Kernel.m	カーネル関数（抽象基底クラス）
KernelGenerator.m	カーネル関数生成器（抽象基底クラス）
KnownSequence.m	潜在状態も既知とする系列。粒子フィルタによる状態系列推定を行わない
LaplaceDistribution.m	ラプラス分布
LikelihoodCalculator.m	系列の尤度を求める抽象既定クラス
MCDCInput.m	モデル推定の入力データ
MCDCMatFile.m	モデル推定の中間ファイル出力クラス
MCDCOutput.m	モデル推定の出力結果
MCDCRegression.m	推定モデルによる回帰プログラム
MCDCStrategy.m	MCMC の動作を定める抽象基底クラス
MCDCStrategyChoice1.m	MCMC の動作を定めるクラス。平均値関数を学習せず、常にアンカーモデルを利用する
MCDCStrategyChoice1_PSMC.m	MCMC の動作を定めるクラス。平均値関数を学習せず、常にアンカーモデルを利用する
MCDCStrategyChoice2.m	MCMC の動作を定めるクラス。現在の GP surface を平均値関数として関数をサンプリングする。
MCDCStrategyChoice3.m	MCMC の動作を定めるクラス。現在の GP surface を平均値関数として関数をサンプリングする。共分散関数を学習せず、固定された共分散行列を用いる
MCDCTest.m	推定モデルの尤度計算プログラム
MCDCTrain.m	モデル推定プログラム
Model.m	状態空間モデルを表す抽象既定クラス
ModelFunctions.m	グリッドの写像に対するスプライン関数
ModelFunctions2.m	グリッドの写像に対するスプライン関数
ModelKind.m	モデル種別
MultipleKnownSequence.m	複数の既知の系列に対する尤度計算を行う
NormalDistribution.m	正規分布
PMCMC.m	モデル推定アルゴリズム。状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定に

ファイル名	内容
	Particle marginal Metropolis-Hastings 法を用いる
PMCMC2.m	モデル推定アルゴリズム。状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定に Particle marginal Metropolis-Hastings 法を用いる。状態空間の特定の一次元のみを共分散関数に用いる
PMCMCParticleFilter.m	粒子フィルタ。PMCMC 法で利用するために Ancestor sampling を行う
PSMC.m	モデル推定アルゴリズム。状態遷移関数と観測関数をガウス過程からのランダムサンプリングで生成する。パラメータ推定は行わない
ParticleFilter.m	粒子フィルタ
PlotGraph.m	グラフ描画サブルーチン。Graphs.m から利用される
RBFKernel.m	RBF カーネル。ガウス過程の共分散関数として利用する
RBFKernelGenerator.m	RBF カーネル関数生成器
RBFKernelGeneratorStrategy.m	RBF カーネル関数の生成方法を定める抽象基底クラス
RBFKernelGeneratorStrategyChoice1.m	RBF カーネル関数生成アルゴリズム。カーネルパラメータを事前分布からランダムサンプリングする
RBFKernelGeneratorStrategyChoice2.m	RBF カーネル関数生成アルゴリズム。カーネルパラメータを現在値からのランダムウォークによって生成する
SimpleSpline.m	一次元でのスプライン補間関数。spline 関数を用いる
SimpleSplineInGrid.m	一次元の状態空間における状態遷移関数用のスプライン補間関数。spline 関数を用いる。遷移先がグリッドの範囲外となった場合にはグリッド端点を値とする
SkewTDistribution.m	Skewed-t 分布
TDistribution.m	t 分布
TLocationScaleDistribution.m	t 位置スケール分布
TruncatedDistribution.m	下限上限で打ち切られた確率分布

ファイル名	内容
UniformDistribution.m	一様分布
VectorValuedFunction.m	各次元のスカラー値関数をベクトル値関数にまとめるルーチン
WeibullDistribution.m	ワイブル分布
logmvnpdf.m	多変量正規分布の確率密度関数の対数を計算するルーチン。Benjamin Dichter により実装され、BSD ライセンスで公開されているもの。

また、samples ディレクトリ以下には、サンプルデータに対して、MCDC ツールを用いてモデル推定、判別実験を行うためのプログラムファイルが含まれています。samples ディレクトリに含まれるファイルの一覧は以下のとおりです。

ファイル名	内容
KitagawaModel.m	Kitagawa モデルによる時系列生成関数
KitagawaModelPMCMC2Estimation.m	Kitagawa モデル推定実験プログラム (PMCMC2 アルゴリズムを利用)
KitagawaModelPMCMC2Estimation_knownSequence.m	Kitagawa モデル推定実験プログラム。潜在状態系列が既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
KitagawaModelPMCMC2Estimation_nonGaussianNoise.m	Kitagawa モデル推定実験プログラム。非ガウスノイズを用いる場合のサンプル (PMCMC2 アルゴリズムを利用)
KitagawaModelPMCMC2Estimation_obsFixed.m	Kitagawa モデル推定実験プログラム。観測モデルが既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
KitagawaModelPMCMC2Estimation_stateFixed.m	Kitagawa モデル推定実験プログラム。状態遷移モデルが既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
KitagawaModelPMCMCEstimation.m	Kitagawa モデル推定実験プログラム (PMCMC アルゴリズムを利用)
KitagawaModelPSMCEstimation.m	Kitagawa モデル推定実験プログラム (PSMC アルゴリズムを利用)
KitagawaModel_WriteGraphs.m	Kitagawa モデル推定結果描画プログラム
LinearStateSpaceModel.m	線形状態空間モデルによる時系列生成関数
LinearStateSpaceModelPMCMC2Estimation.m	線形状態空間モデル推定実験プログラム (PMCMC2 アルゴリズムを利用)
LinearStateSpaceModelPMCMCEstimation.m	線形状態空間モデル推定実験プログラム (PMCMC アルゴリズムを利用)

ファイル名	内容
LinearStateSpaceModelPSMCEstimation.m	線形状態空間モデル推定実験プログラム (PSMC アルゴリズムを利用)
LorenzModel.m	Lorenz モデルによる時系列生成関数
LorenzModelPMCMC2Estimation.m	Lorenz モデル推定実験プログラム (PMCMC2 アルゴリズムを利用)
LorenzModelPMCMC2EstimationWithoutAnchor.m	Lorenz モデル推定実験プログラム。アンカーモデルを利用しない場合のサンプル (PMCMC2 アルゴリズムを利用)
LorenzModelPMCMC2Estimation_knownSequence.m	Lorenz モデル推定実験プログラム。潜在状態系列が既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
LorenzModelPMCMC2Estimation_obsFixed.m	Lorenz モデル推定実験プログラム。観測モデルが既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
LorenzModelPMCMC2Estimation_stateFixed.m	Lorenz モデル推定実験プログラム。状態遷移モデルが既知の場合のサンプル (PMCMC2 アルゴリズムを利用)
LorenzModelPMCMCEstimation.m	Lorenz モデル推定実験プログラム (PMCMC アルゴリズムを利用)
LorenzModelPSMCEstimation.m	Lorenz モデル推定実験プログラム (PSMC アルゴリズムを利用)
LorenzModel_WriteGraphs.m	Lorenz モデル推定結果描画プログラム
MotionCapture.m	MotionCapture データからの時系列データ生成関数
MotionCapturePMCMC2Estimation.m	MotionCapture モデル推定実験プログラム (PMCMC2 アルゴリズムを利用)
MotionCapturePMCMC2Test.m	MotionCapture クラス分類実験プログラム
MotionCapture_WriteGraphs.m	MotionCapture モデル推定結果描画プログラム
amc_to_matrix.m[3]	AMC ファイルから Matlab 行列形式への変換関数

[3] CMU Graphics Lab Motion Capture Database (http://mocap.cs.cmu.edu/) にて公開されているものです。プログラムの動作に必要となるため同梱しています。

4. ライセンス表記

MCDC ツールの実験用サンプルデータのうち、モーションキャプチャデータを用いるものについては、以下のウェブサイトにて公開されているデータ、ツールを利用して実行します。

CMU Graphics Lab Motion Capture Database

```
http://mocap.cs.cmu.edu/
```

ウェブサイトに記載されている利用許諾条件を以下に示します。

```
This data is free for use in research projects.
You may include this data in commercially-sold products,
but you may not resell this data directly, even in converted form.
If you publish results obtained using this data, we would appreciate it
if you would send the citation to your published paper to jkh+mocap@cs.cmu.edu,
and also would add this text to your acknowledgments section:
The data used in this project was obtained from mocap.cs.cmu.edu.
The database was created with funding from NSF EIA-0196217.
```