Monte Carlo Dynamic Classifier

Reference manual

Version 1.2.0
29/03/2016

ChangeLog

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

Contents

1. **INTRODUCTION** **1**

2. **PROGRAM EXECUTION** **2**

2.1. MCDCTrain 2
2.1.1. Execution Method 2
2.1.2. Parameter 3
2.1.3. Return value 4
2.1.4. Intermediate file 5
2.1.5. Log file 5
2.2. MCDCTest 6
2.2.1. Execution Method 6
2.2.2. Parameters 6
2.2.3. Return Values 6
2.3. Graphs 7
2.3.1. Graphs.YE 7
2.3.2. Graphs.XE 7
2.3.3. Graphs.YEMean 8
2.3.4. Graphs.XEMean 8
2.3.5. Graphs.Loglik 9
2.3.6. Graphs.Rmse 10
2.4. Example of Program Execution 10
2.4.1. Generation of Observed Data 10
2.4.2. Design of State Space 11
2.4.3. Model Design 11
2.4.4. Configuration of Algorithm of MCDCTrain 12
2.4.5. Configuration as to execution of model estimation 12
2.4.6. Execution of model estimation 13
2.5. Advanced use of MCDCTrain 13
2.5.1. Given state function 14
2.5.2. Given observation function 14
2.5.3. Given state sequence 14
2.5.4. Estimation of the state sequence with non-Gaussian noise 15
2.5.5. Use of multiple sequences 16
2.5.6. Estimation with Metropolis-Hastings with Gibbs sampling 17

3. **PROGRAM STRUCTURE** **18**

3.1. File Structure 18

4. **LICENSE** **24**

1. Introduction

This manual is to explain the program execution procedures created in the "Monte Carlo Dynamic Classifier (MCDC) Tools development and experiment supporting work". Monte Carlo Dynamic Classifier Tools is a program that performs model estimation of arbitrary observed data sequences and estimation of state sequences of its estimated model. The estimated model can be used for class separation of observed data sequences by applying it to different observed data sequences and calculating the likelihood of the model. MCDC Tools is composed of the following program bundle:

MCDCTrain

Model estimation program

MCDCTest

Calculation of model likelihood program

Graphs

A set of functions for graph drawing of estimated models

The following chapters of this manual explain the execution methods and examples of these programs (Chapter 2), and program structures (Chapter 3). Further, in a section of the sample program attached to these tools, motion capture data is used. The motion capture data is offered publicly in the CMU Graphics Lab Motion Capture Database. The use permission conditions are stated in Chapter 4.

2. Program Execution

MCDC Tools offers MCDC Train that performs model estimation for observed data sequences, and MCDC Test, which conduct an estimation of state sequences of unknown observed data sequences using the estimated model.

Additionally, Graphs class is also offered as a tool of compiled sets of functions that draws graphs of estimated models. In this chapter, the execution methods of these programs are explained.

2.1. MCDCTrain

MCDC Train function is a program, which performs model estimation for observed data sequences.

2.1.1. Execution Method

MCDC Train function is executed as follows：

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

It can also read the output file from past executions and perform continuous iterative executions. In this case, the MCDC Train function is executed as follows. In case of iterative executions, read the parameters from the designated MAT-File and resume execution with the same configurations as previously. However, by assigning a class of Name and Value, the previous configuration can be overwritten and executed.

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

2.1.2. Parameter

MCDC Train function parameters are as follows:

Parameter	Data type	Description
algorithm	Algorithm	Algorithm Classification.
xGrids	G double[] cell	Coordinates of lattice point formed on state space. Display each dimensional value as cell array stored as double array.
stateKernelGens	Dx handle cell	Kernel prior distribution of state transition function. Dx indicates the number of dimensions subject to estimation in state space.
obsKernelGens	P handle cell	Kernel prior distribution of observation function.
gridDimForGramMatrix	int	Dimensions used to create gram matrix.
stateSplineHandle	handle	Algorithm used for spline interpolation for the state function.
obsSplineHandle	handle	Algorithm used for spline interpolation for the observation function.
x0	Dx*1 double	Initial state of state variables.
xaux	Da*T double	Additional state data.
u	D*T double	Control data.
z	P*T double	Observed data.
N	int	Number of particles when particle filter is executed.
J	int	Entire number of MCMC iterations.
K	int	Observed offset.
aspect	Aspect	System configuration information (log output destination, etc.).

Select algorithm parameter from any of the classes below:

Class name	Description
PSMC	Generate state transition function and observed function by random sampling from Gaussian process. This does not perform parameter estimation.
PMCMC	Generate state transition function and observed function by random sampling from Gaussian process. Use particle marginal Metropolis-Hastings (PMCMC) method for parameter estimation.
PMCMC2	Generate state transition function and observed function by random sampling from Gaussian process. Use particle marginal Metropolis-Hastings method for parameter estimation. Use one dimension only specific to state space for

Class name	Description
	covariance function.

When PMCMC2 is used as algorithm, select a learning method of mean value function and covariance function, or a learning method of kernel parameters used in covariance function from below:

Class name	Description
MCDCStrategyChoice1	Without learning average function, always use anchor model. Use kernel function for covariance function.
MCDCStrategyChoice2	Perform sampling of functions from present GP surface used as a mean value function. Use kernel function for covariance function.
MCDCStrategyChoice3	Perform sampling of functions from present GP surface used as a mean value function. Use fixed covariance matrix without using kernel function for covariance function.
RBFKernelGeneratorStrategyChoice1	Perform a random sampling of RBF kernel's kernel-parameters from prior distribution.
RBFKernelGeneratorStrategyChoice2	Generate RBF kernel's kernel-parameters from present value of random walk.

The handle of an own designed spline function can be passed. The default spline functions are as follows:

Function	Description
GenericSpline	General spline function using "spapi" for the multiple dimensional state space.
SimpleSpline	Simple and fast spline function using "spline" for 1-dimensional state space.
SimpleSplineInGrid	Simple and fast spline function for 1-dimensional state space.　When the transition destination is outside the grid, it is pulled back to the end point of the grid.

2.1.3. Return value

Returned values as a result of MCDC Train function are as follows:

Value	Data type	Description
IDX	J*1 double	Number of cumulative receipts until (j)th iteration.
SKP	A*D*2 double	At (a)th received iteration [1], kernel parameters of state transition function (sigma, l).

Value	Data type	Description
OKP	A*P*2 double	At (a)th received iteration, kernel parameters of observed function (sigma, l).
FV	A*D*G double	At (a)th received iteration, values on lattice point by state transition function. G indicates a multidimensional array that corresponds with the size of lattice point.
GV	A*P*G double	At (a)th received iteration, values on lattice point by observation function. G indicates a multidimensional array that corresponds with the size of lattice point.
XE	A*T*D double	At (a)th received iteration, estimated mean of state variables at time (t).
YE	A*T*P double	At (a)th received iteration, estimated mean of observed variables at time (t).
loglik	A*1 double	At (a)th received iteration, estimated logarithmic likelihood by particle filter.

2.1.4. Intermediate file

Intermediate state will be stored in accordance with the aspect settings when executing. This will be output as a dump file in .MAT format. The data stored in the file is as follows:

Value	Data type	Description
algorithm	Algorithm	Algorithm passed to MCDC Train parameters.
in	MCDCInput	All MCDC Train parameters excluding algorithm and aspect.
out	MCDCOutput	All return values of MCDC Train (halfway state) and present number of iteration (j).

2.1.5. Log file

The log file will be outputted in accordance with the aspect settings when executing. This is a text format file. The log file will be the following format.

```
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

MCDC Test function estimates unknown state of data using the acquired model by MCDC Train. This can be applied to a class separation problem by performing state estimation using multiple different models and comparing the likelihood of each.

2.2.1. Execution Method

MCDC Test function is executed as follows.

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

2.2.2. Parameters

Parameters of MCDC Test function are as follows:

Parameter	Data type	Description
u	D*T double	Control data.
y	P*T double	Observed data.
N	int	Number of particles when particle filter is executed.
modelFile	chars	Model file.

2.2.3. Return Values

Returned values as a result of MCDC Test function are as follows:

Value	Data type	Description
result	ParticleFilter	Number of cumulative receipts up to (j)th iteration.
FnState	handle	State transition function.
FnObs	handle	Observation function.

The result of return value includes the result of state estimation by particle filter, which has the following structures:

Value	Data type	Description
Particles	N*T*D double	Coordinates of each particle at time (t).
Weights	N*T double	Weight of each particle at time (t).
Loglik	double	Logarithmic likelihood of estimated state.

2.3. Graphs

The model estimated by MCDC Train can be outputted into a PDF file using a set of functions defined in Graph class.　The following graphs can be outputted.

2.3.1. Graphs.YE

This outputs a temporal transition graph of results by pursuing observed data by particle filters using observed data sequences and estimated models.　In MCMC iteration, it outputs the results using an estimated model at designated specific iterations.　It also outputs graphs with respect to each dimension of observed data sequences.　It is executed as follows:

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

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by the number of dimensions and extension of .PDF.
matFileName	char[]	MATLAB data file name, including estimated model.
iterations	int[]	Matrix that lists MCMC iteration of plotted estimated values.
times	int[]	Range of plotted data sequence.　Output the entire data sequence when omitted.

2.3.2. Graphs.XE

It outputs a temporal transition graph of state sequences estimated values by particle filters using estimated models.　In MCMC iteration, it outputs a result using an estimated model at a designated specific iteration.　It also outputs a graph with respect to each dimension of state sequences.　It is executed as follows.　It is executed as follows.

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

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by the number of dimensions and extension of .PDF.
matFileName	char[]	MATLAB data file name, including estimated model.
iterations	int[]	Matrix that lists MCMC iteration of plotted estimated values.
times	int[]	Range of plotted data sequence.　Output the entire data sequence when omitted.

2.3.3. Graphs.YEMean

This outputs a temporal transition graph of results by pursuing observed data by particle filters using observed data sequences and estimated models.　It uses the mean estimated model of entire MCMC iteration.　It also outputs graphs with respect to each dimension of observed data sequences.　It is executed as follows.

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

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by the number of dimensions and extension of .PDF.
matFileName	char[]	MATLAB data file name, including estimated model.
times	int[]	Range of plotted data series.　Output the entire data sequence when omitted.

2.3.4. Graphs.XEMean

This outputs a temporal transition graph of state sequence of estimated values by particle filters using estimated models. It uses the mean estimated model of the entire MCMC iteration. It also outputs a graph with respect to each dimension of state sequence. It is executed as follows.

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

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by the number of dimensions and extension of .PDF.
matFileName	char[]	MATLAB data file name, including estimated model.
times	int[]	Range of plotted data series. Output the entire data sequence when omitted.

2.3.5. Graphs.Loglik

In model estimation, it outputs logarithmic likelihood values estimated by each MCMC iteration, which is executed as follows.

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

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by extension of .PDF.
matFileName1, 2, ...	char[]	MATLAB data file name, including estimated model. If multiple files are selected, it draws graphs of each individually as one sequence.

The return value is as follows:

Value	Data type	Description
loglik	double	Log-likelihood value

2.3.6. Graphs.Rmse

In model estimation, it outputs mean square error of the results of estimated observed data sequences using estimated models up to each MCMC iteration, which is executed as follows.

```
Graphs.Rmse( ...
  outputFileNamePrefix, ...
  matFileName ...
);
```

Parameters are as follows:

Parameter	Data type	Description
outputFileNamePrefix	char[]	File name prefix of output graph file. The file name will be a designated prefix followed by extension of .PDF.
matFileName	char[]	MATLAB data file name, including estimated model.

The return values are as follows

Value	Data type	Description
RmseMean	double	The mean of the mean square errors at each MCMC iteration.
RmseStd	double	The standard deviation of the mean square errors at each MCMC iteration.
Rmse	double[]	The matrix of the mean square errors at each MCMC iteration.

2.4. Example of Program Execution

As an execution example of model estimation by MCDC Train, the performing of estimation of observed data sequences generated from Kitagawa's model is show. The code samples are included in the following folder.

```
samples/KitagawaModelPMCMC2Estimation.m
```

2.4.1. Generation of Observed Data

First, prepare observed data subject to model estimation. Generally, observed data should be provided in advance, here we use data sequences generated from Kitagawa's model as observed data.

```
[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)';
```

Due to this code, observed data sequences are stored in y. State sequences are stored in x, but x is not going to be used after this process. Further, in Kitagawa's model, to give time-varying control data, control data sequences are also generated here together with observed data.

2.4.2. Design of State Space

MCDC Train performs model estimations with state space models being unknown. With the current program, the number of dimensions of state space and the moving range for values of state variables need to be given. In the following, consider the two-dimensional state space and configure the lattice point in the range from -30 to 30 at 2.0 increments for each dimension.

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

In the process of model estimation, perform sampling a value of function from the Gaussian process on the lattice point configured here and by performing spline interpolation so that state transition function and observation function are created. The expressive power of the model is stronger when many lattice points are selected in a wider extent. However, it will significantly increase the processing time and the amount of memory used.

2.4.3. Model Design

Next, design models for state transition function and observed function. In MCDC Train, mean value function when sampling functions from Gaussian process and kernel covariance matrix can be given to both state transition function and observation function.

```
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)  ...
};
```

In the example codes above, use RBF kernel conforming to uniform distribution with parameters, σ and l, both being [0.01, 10], and configure Kitagawa's model state transition function and observed function to mean value function[2].

2.4.4. Configuration of Algorithm of MCDCTrain

Select Algorithm used for MCMC iteration of MCDC Train. Choose either fixing kernel parameters and mean value function in prior distribution or transit it every time.

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

For MCMC iteration, the MCMC algorithm is set as follows. The argument value of 1.0 means that the MH with Gibbs algorithm is not (refer section 2.5).

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

When both state and observation functions are unknown and estimated with GP, the following settings are specified. When state or observation function is known, the settings in sections 2.5.1 or 2.5.2 are specified.

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

For MCDCStrategyChoice2, a dimension in the state space which can be dependent with the other dimensions is selected and used to boost the MCMC iterations. GenericSpline for multi-dimensional interpolation should be specified in this case.

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

2.4.5. Configuration as to execution of model estimation

Configure the number of iterations of model estimation, number of particles and

[2]In general, a true model is unknown, configuring a true model as mean value function cannot be set.

output destination of log files.

```
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;
```

When the state sequence is unknown and estimated by the particle filter algorithm, the following settings are specified. When the state sequence is known or multiple sequences are available, the settings in section 2.5.3 are specified.

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

2.4.6. Execution of model estimation

Use all the configurations up to here and execute model estimation by MCDCTrain.

```
[ IDX, SKP, OKP, FV, GV, XE, YE, loglik ] = MCDCTrain( ...
  algorithm, …
  grids, ...
  stateKernelGens, ...
  obsKernelGens, ...
  stateMeanFuncs, ...
  obsMeanFuncs, ...
  gridDimForGramMatrix, ...
  splineHandle, ...
  x0, ...
  [], ... % No auxiliary states
  u', ...
  y', ...
  N, ...
  J, ...
  0, ...
  aspect ...
);
```

2.5. Advanced use of MCDCTrain

When executing MCDCTrain, some alternative procedures are available. Here the settings are described.

2.5.1. Given state function

For MCDCTrain, when the state function is known, the settings are specified by using SetStateModel. In such a case, only the observation function is estimated. An example is as follows:

```
% Define the state function as an array.
stateMeanFuncs = { ...
  @(x) (a1 .* x(1,:) + b1 .* x(1,:) / (1 + x(1,:) .^ 2)), ...
  @(x) (a2 .* x(2,:) + b2 .* x(2,:) / (1 + x(2,:) .^ 2))  ...
};

% Generate the algorithm object.
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Set the state model and the algorithm object.
stateModel = FixedModel(ModelKind.State, VectorValuedFunction(stateMeanFuncs));
algorithm.SetStateModel(stateModel);
```

The detailed information is in the following folder.

```
samples/KitagawaModelPMCMCEstimation_stateFixed.m
```

2.5.2. Given observation function

For MCDCTrain, when the observation function is known, the settings are specified using SetObsmodel. In such a case, only the state function is estimated. An example is as follows.

```
% Define the observation function as an array.
obsMeanFuncs = { ...
  @(x) (d1 .* x(1,:) .^ 2 + d2 .* x(2,:) .^ 2), ...
  @(x) (d3 .* x(1,:) .^ 2                   ), ...
  @(x) (                    d4 .* x(2,:) .^ 2)  ...
};

% Generate the algorithm object.
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Set the observation model and the algorithm object.
obsModel = FixedModel(ModelKind.Observation, VectorValuedFunction(obsMeanFuncs));
algorithm.SetObsModel(obsModel);
```

The detailed information is in the following folder.

```
samples/KitagawaModelPMCMCEstimation_obsFixed.m
```

2.5.3. Given state sequence

For MCDCTrain, when the state sequence is known, the acceptance probability for each MCMC iteration is calculated using the pair of the observation and state sequences without the particle filter algorithm which is used to estimate the state sequence in the case that the state sequence is unknown. An example is shown as follows:

```
% Generate the algorithm object.
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Set the state sequence and the algorithm object.
likelihoodCalculator = KnownSequence(x, xaux, u, y, q, r, K);
algorithm.SetLikelihoodCalculator(likelihoodCalculator);
```

The parameters in the KnownSequence class are as follows:

Parameter	Data type	Description
x	Dx*T double	State data
xaux	Da*T double	Additional state data
u	D*T double	Control data
y	P*T double	Observed data
q	double	Variance of state noise
r	double	Variance of observed noise
K	int	Observed offset

The detailed information in the following folder.

```
samples/KitagawaModelPMCMCEstimation_knownSequence.m
```

2.5.4. Estimation of the state sequence with non-Gaussian noise

For the model estimation with MCDCTrain, Gaussian noise is assumed as default. However, several noises can be specified for the algorithm object using SetStateNoise. An example is as follows:

```
% Generate the algorithm object.
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Generate the noise distribution for the state sequence.
xdim = size(x0, 2);
stateNoise = CauchyDistribution(0, q, xdim);

% Set the noise distribution to the algorithm object.
algorithm.SetStateNoise(stateNoise);
```

For the observation sequence, the noise is specified using SetObsNoise in a similar way.

Parameters are as follows. “dim” means the dimension of the state or observation data.

Noise distribution and parameters	Description

Noise distribution and parameters	Description
CauchyDistribution(m,v,dim)	Cauchy distribution (Mean m, Variance v)
ExponentialDistribution(mu,dim)	Exponential distribution (Mean mu)
GammaDistribution(a,b,dim)	Gamma distribution (Shape a, Scale b)
GeneralizedParetoDistribution(k,sigma,theta,dim)	Generalized Pareto distribution (Shape k, Scale sigma, Location theta)
LaplaceDistribution(location,scale,dim)	Laplace distribution (Location location, Scale scale)
NormalDistribution(m,v,dim)	Normal distribution (Mean m, Variance v)
TDistribution(nu,dim)	t distribution (Degree of freedom nu)
TLocationScaleDistribution(m,v,nu,dim)	t location-scale distribution (Mean m, Variance v, Degree of freedom nu)
UniformDistribution(lb,ub,dim)	Uniform distribution (Lower lb, Upper ub)
WeibullDistribution(a,b,dim)	Weibull distribution (Scale a, Shape b)

The detailed information is in the following folder.

```
samples/KitagawaModelPMCMCEstimation_nonGaussianNoise.m
```

2.5.5. Use of multiple sequences

An example for use of multiple sequences is as follows:

```
% Generate the algorithm object.
algorithm = PMCMC2(kernelGeneratorStrategy, mcdcStrategy);

% Set the multiple sequences to the algorithm object.
likelihoodCalculator = MultipleKnownSequence(X, xaux, u, Y, q, r, K);
algorithm.SetLikelihoodCalculator(likelihoodCalculator);
```

The parameters for the MultipleKnownSequence class are as follows:

Parameter	Data type	Description
X	Cell of Dx*T double	Cell array for state data Dx*T matrix for each cell
xaux	Cell of Da*T double	Cell array for additional state data Da*T matrix for each cell
u	Cell of D*T double	Cell array for control data D*T matrix for each cell

Parameter	Data type	Description
Y	Cell of P*T double	Cell array for observed data P*T matrix for each cell
q	double	Variance of sate noise
r	double	Variance of observed noise
K	int	Observed offset

2.5.6. Estimation with Metropolis-Hastings with Gibbs sampling

In each EM iteration, either PMCMC or MH with Gibbs sampling algorithm is selected with a pre-fixed probability.　The probability of 0 means that MH with Gibbs sampling is always selected, while the probability of 1 means that PMCMC is always selected. The default is 0.25.

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

3. Program Structure

This chapter explains the program structure of MCDC Tools.

3.1. File Structure

MCDC Tools is created as a MAT-Lab program. The list of program files is as follows:

File name	Description
Algorithm.m	Abstract base class that represents algorithm used for model estimation.
Aspect.m	Configuration class to set program operation.
BoundedNormalDistribution.m	Normal distribution limited within positive range.
CheckGridTransformation.m	Ranges check function for grid mapping.
CompositeLikelihoodCalculator.m	Log-likelihood calculation for multiple sequences
ContinuousUnivariateDistribution.m	Continuous univariate distribution
DBA.m	DTW Barycenter Averaging
DBAAlign.m	Align the lengths of the multiple sequences with DBA
Distribution.m	Abstract base class that represents probability distribution.
ExponentialDistribution.m	Exponential distribution.
FixedModel.m	Fixed model without estimation
FixedValueDistribution.m	Distribution with a fixed value
GPSurface.m	Function that requires grid mapping.
GammaDistribution.m	Gamma distribution
GaussianProcessModel.m	State space model with GP estimation
GaussianProcessModelRotate.m	State space model with GP estimation. A dimension in the state space is selected and used to boost the MCMC iterations.
GeneralizedParetoDistribution.m	Generalized Pareto distribution
GenericSpline.m	Multidimensional spline interpolation function. Use “spapi” function.
Graphs.m	Graph drawing function group.
GridData.m	Grid structure in state space.
IterationStrategy.m	EM iteration strategy
IterationStrategyDefault.m	Use of PMCMC
IterationStrategyMHGibbs.m	Use of MH with Gibbs sampling

File name	Description
IterationStrategyProbabilistic.m	Selectively use of PMCMC and MH with Gibbs sampling with a pre-fixed probability.
Kernel.m	Kernel function (abstract base class).
KernelGenerator.m	Kernel function generator (abstract base class).
KnownSequence.m	Use of known sequence.
LaplaceDistribution.m	Laplace distribution.
LikelihoodCalculator.m	Likelihood calculation.
MCDCInput.m	Input data of model estimation.
MCDCMatFile.m	Intermediate file output class of model estimation.
MCDCOutput.m	Output result of model estimation.
MCDCRegression.m	Regression using the estimated model.
MCDCStrategy.m	Abstract base class that determines MCMC's operations.
MCDCStrategyChoice1.m	Class that determines MCMC's operations.　Without learning mean value function, always use anchor model.
MCDCStrategyChoice1_PSMC.m	Class that determines MCMC's operations.　Without learning mean value function, always use anchor model.
MCDCStrategyChoice2.m	Class that determines MCMC's operations.　Perform sampling of functions from present GP surface used as mean value function.
MCDCStrategyChoice3.m	Class that determines MCMC's operations. Perform sampling of functions from present GP surface used as mean value function. Without learning covariance function, use fixed covariance matrix.
MCDCTest.m	Likelihood calculation program of estimation model.
MCDCTrain.m	Model estimation program.
ModelFunctions.m	Spline function for grid mapping.
ModelFunctions2.m	Spline function for grid mapping.
ModelKind.m	Model kind
MultipleKnownSequence.m	Calculation for multiple known sequences
NormalDistribution.m	Normal distribution.
PMCMC.m	Model estimation algorithm.　Generate

File name	Description
	state transition function and observed function by random sampling from Gaussian process. Use particle marginal Metropolis-Hastings (PMCMC) method for parameter estimation.
PMCMC2.m	Model estimation algorithm. Generate state transition function and observed function by random sampling from Gaussian process. Use particle marginal Metropolis-Hastings method for parameter estimation. Use one dimension only specific to state space for covariance function.
PMCMCParticleFilter.m	Particle filter. Perform ancestor sampling to use it by PMCMC method
PSMC.m	Model estimation algorithm. Generate state transition function and observed function by random sampling from Gaussian process. Parameter estimation is not performed.
ParticleFilter.m	Particle filters.
PlotGraph.m	Graph drawing subroutines. Used from Graphs.m.
RBFKernel.m	RBF kernel. Use it as a covariance function of the Gaussian process.
RBFKernelGenerator.m	RBF kernel function generator.
RBFKernelGeneratorStrategy.m	Abstract base class that determines a generating method for RBF kernel function.
RBFKernelGeneratorStrategyChoice1.m	RBF kernel function generating algorithm. Perform random sampling of kernel parameters from prior distribution.
RBFKernelGeneratorStrategyChoice2.m	RBF kernel function generating algorithm. Generate kernel parameters by random walk from present value.
SimpleSpline.m	Simple spline function for 1-dimensional state space with “spline” function.
SimpleSplineInGrid.m	Simple spline function for 1-dimensional state space with “spline” function. When the transition destination is outside the grid, it is pulled back to the end point of

File name	Description
	the grid.
SkewTDistribution.m	Skewed-t distribution
TDistribution.m	t distribution
TLocationScaleDistribution.m	t location-scale distribution
TruncatedDistribution.m	Distribution truncated by lower and upper limits.
UniformDistribution.m	Uniform distribution.
VectorValuedFunction.m	Routines that compile scalar valued functions of each dimension to vector valued functions.
WeibullDistribution.m	Weibull distribution
logmvnpdf.m	Log multivariate normal distribution function by Benjamin Dichter with BSD license.

Also, the following sample directories contain program files to perform model estimation using MCDC Tools and conduct discrimination experiments for sample data. The list of files included in the sample directories are as follows:

File name	Description
KitagawaModel.m	Time series generating function by Kitagawa's model.
KitagawaModelPMCMC2Estimation.m	Kitagawa's model estimation experiment program (use PMCMC2 algorithm).
KitagawaModelPMCMC2Estimation_knownSequence.m	Kitagawa's model estimation experiment program with a known state sequence. (use PMCMC2 algorithm)
KitagawaModelPMCMC2Estimation_nonGaussianNoise.m	Kitagawa's model estimation experiment program with non-Gaussian noise. (use PMCMC2 algorithm)
KitagawaModelPMCMC2Estimation_obsFixed.m	Kitagawa's model estimation experiment program with a known observed sequence. (use PMCMC2 algorithm)
KitagawaModelPMCMC2Estimation_stateFixed.m	Kitagawa's model estimation experiment program with a given state function. (use PMCMC2 algorithm)
KitagawaModelPMCMCEstimation.m	Kitagawa's model estimation

File name	Description
	experiment program (use PMCMC Algorithm).
KitagawaModelPSMCEstimation.m	Kitagawa's model estimation experiment program (use PSMC algorithm).
KitagawaModel_WriteGraphs.m	Kitagawa's model estimation result drawing program.
LinearStateSpaceModel.m	Time series generating function by linear state space model.
LinearStateSpaceModelPSMCEstimation.m	Linear state space model estimation experiment program (use PSMC algorithm).
LinearStateSpaceModelPMCMC2Estimation.m	Linear state space model estimation experiment program (use PMCMC2 algorithm).
LinearStateSpaceModelPMCMCEstimation.m	Linear state space model estimation experiment program (use PMCMC algorithm).
LorenzModel.m	Time series generating function by Lorenz's model.
LorenzModelPMCMC2Estimation.m	Lorenz's model estimation experiment program (use PMCMC2 algorithm).
LorenzModelPMCMC2EstimationWithoutAnchor.m	Lorenz's model estimation experiment program without anchor model. (use PMCMC2 algorithm)
LorenzModelPMCMC2Estimation_knownSequence.m	Lorenz's model estimation experiment program with a known state sequence. (use PMCMC2 algorithm)
LorenzModelPMCMC2Estimation_obsFixed.m	Lorenz's model estimation experiment program with a given observation function. (use PMCMC2 algorithm)
LorenzModelPMCMC2Estimation_stateFixed.m	Lorenz's model estimation experiment program with a given state function. (use PMCMC2 algorithm)
LorenzModelPMCMCEstimation.m	Lorenz's model estimation experiment program (use

File name	Description
	PMCMC algorithm).
LorenzModelPSMCEstimation.m	Lorenz's model estimation experiment program (use PSMC algorithm).
LorenzModel_WriteGraphs.m	Lorenz's model estimation result drawing program.
MotionCapture.m	Time series generating function from motion capture data.
MotionCapturePMCMC2Estimation.m	Motion capture model estimation experiment program (use PMCMC2 algorithm).
MotionCapturePMCMC2Test.m	Motion capture class separation experiment program.
MotionCapture_WriteGraphs.m	Motions capture model estimation result drawing program.
amc_to_matrix.m[3]	Transfer function from AMC file to MATLAB matrix format.

[3] This code is originally contained in the CMU Graphics Lab Motion Capture Database (http://mocap.cs.cmu.edu/). This is required for the execution of the program.

4. License

Among the experimental sample data in MCDC tools, the samples that use motion capture data are executed using data and tools that are published on the following website.

CMU Graphics Lab Motion Capture Database

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

Use permission conditions are posted on the website shown below.

```
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.
```