AKalmanLibrary

1.0

Generated by Doxygen 1.8.15

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 ROUKF Class Reference

`#include <ROUKF.h>`

Public Types

- enum PARAM_TYPE { **DEFAULT**, **POSITIVE**, **RANGED_LOG_DIST**, **RANGED_NORMAL_DIST** }

Public Member Functions

- ROUKF (int nObservations, int nStates, int nParameters, double ∗statesUncertainty, double ∗parameters↩ Uncertainty, SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution)
- double executeStep (double ∗Zkhatc, forwardOp A, observationOp H)
- double executeStepParallel (double ∗Zkhatc, forwardOp A, observationOp H, int seed, MPI_Comm local_↩ comm, MPI_Comm masters_comm)
- void reset (int nObservations, int nStates, int nParameters, double ∗statesUncertainty, double ∗parameters↩ Uncertainty, SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution)
- void toString ()
- void **getState** (double ∗∗XC)
- void **setState** (double ∗XC)
- void **getParameters** (double ∗∗ThetaC)
- void **setParameters** (double ∗ThetaC)
- vector< double > getParametersStd ()
- void **getError** (double ∗∗err)
- double **getObsError** (int numObservation)
- int **getObservations** () const
- int **getStates** () const

2.1.1 Detailed Description

Implementation of the ROUKF.

Usage:

int (*ptA)(double, int) = NULL; void (*ptH)(double, int, double*, int) = NULL; ptA = ptH =

Initialization parameters for (int i = 0; i < nParameters; i++) { initialGuess[i] = 2.5E5; // -> X0 parameters↩ Uncertainties[i] =1E8; // -> U0^{-1} } ROUKF *kalmanInstance = new ROUKF(nStates, nParameters, states↩ Uncertainties, parametersUncertainties);

Set initial condition kalmanInstance->setState(initialGuess, nParameters);

for (int it = 0; it < 3000; it++) { error = kalmanInstance->executeStep(observation, nStates, ptA, ptH); }

Get the Kalman estimation kalmanInstance->getState(&sol); // -> XSol Class that implements the reduced order unscented Kalman filter.

2.1.2 Member Enumeration Documentation

2.1.2.1 PARAM_TYPE

```
enum ROUKF::PARAM_TYPE
```

Not implemented yet.

2.1.3 Constructor & Destructor Documentation

2.1.3.1 ROUKF()

```
ROUKF::ROUKF (
            int nObservations,
            int nStates,
            int nParameters,
            double * statesUncertainty,
            double * parametersUncertainty,
            SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution )
```

Creates the covariance matrixes and sigma points associated with the extended vector X and their uncertainty.

Parameters

nObservations	Quantity of observations.
nStates	Quantity of states.
nParameters	Quantity of parameters.
statesUncertainty	Vector with the uncertainty of each state in X.
parametersUncertainty	Vector with the uncertainty of each parameter in Theta.
sigmaDistribution	Type of sigmas applied to assess the unscented transform. Only SIMPLEX is available by now.

2.1.4 Member Function Documentation

2.1.4.1 executeStep()

```
double ROUKF::executeStep (
            double * Zkhatc,
            forwardOp A,
            observationOp H )
```

Performs one step of the Kalman filtering process in serial execution of the sigma points.

Parameters

Zkhatc	Current observations estimations.
A	Forward operator.
H	Observation operator;

Returns

Current L2 norm of the errors across all observations.

2.1.4.2 executeStepParallel()

```
double ROUKF::executeStepParallel (
            double * Zkhatc,
            forwardOp A,
            observationOp H,
            int seed,
            MPI_Comm local_comm,
            MPI_Comm masters_comm )
```

Performs one step of the Kalman filtering process with parallel execution of the sigma points.

Parameters

Zkhatc	Current observations estimations.
A	Forward operator.
H	Observation operator;
seed	Sigma point ID for the current MPI process.
local_comm	Communicator of all MPI processes that solve the sigma point `seed`.
masters_comm	Communicator of the master MPI processes of each sigma point `seed`.

Returns

Current L2 norm of the errors across all observations.

2.1.4.3 getParametersStd()

```
vector< double > ROUKF::getParametersStd ( )
```

Returns a vector with the standard variation of each parameter at the current iteration.

Returns

Vector with the standard variation of each parameter at the current iteration.

2.1.4.4 reset()

```
void ROUKF::reset (
            int nObservations,
            int nStates,
            int nParameters,
            double * statesUncertainty,
            double * parametersUncertainty,
            SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution )
```

Returns to the initial state of the kalman filter. Not fully tested

Parameters

nObservations	Quantity of observations.
nStates	Quantity of states.
nParameters	Quantity of parameters.
statesUncertainty	Vector with the uncertainty of each state in X.
parametersUncertainty	Vector with the uncertainty of each parameter in Theta.
sigmaDistribution	Type of sigmas applied to assess the unscented transform. Only SIMPLEX is available by now.

2.1.4.5 toString()

```
void ROUKF::toString ( )
```

Prints the private attributes of the ROUKF instance.

The documentation for this class was generated from the following files:

- ROUKF.h
- ROUKF.cpp

2.2 SigmaPointsGenerator Class Reference

`#include <SigmaPointsGenerator.h>`

Public Types

- enum **SIGMA_DISTRIBUTION** { **SIMPLEX**, **CANONIC**, **STAR**, **SIMPLEX_STAR** }

Static Public Member Functions

- static void generateSigmaPoints (int nParameters, SIGMA_DISTRIBUTION distribution, arma::mat ∗sigma)

Static Protected Member Functions

- static void canonicSigmaPoints (int nParameters, arma::mat ∗sigma)
- static void simplexSigmaPoints (int nParameters, arma::mat ∗sigma)
- static void starSigmaPoints (int nParameters, arma::mat ∗sigma)
- static void **simplexStarSigmaPoints** (int nParameters, arma::mat ∗sigma)

2.2.1 Detailed Description

Static class which generates different distributions of sigma points.

2.2.2 Member Function Documentation

2.2.2.1 canonicSigmaPoints()

```
void SigmaPointsGenerator::canonicSigmaPoints (
            int nParameters,
            arma::mat * sigma )  [static], [protected]
```

Generates the canonical sigma points distribution which contains symmetric independent perturbations of the parameters.

2.2.2.2 generateSigmaPoints()

```
void SigmaPointsGenerator::generateSigmaPoints (
            int nParameters,
            SIGMA_DISTRIBUTION distribution,
            arma::mat * sigma )  [static]
```

Switch function that invokes the appropriate function for the sigma points generation.

2.2.2.3 simplexSigmaPoints()

```
void SigmaPointsGenerator::simplexSigmaPoints (
            int nParameters,
            arma::mat * sigma )  [static], [protected]
```

Generates the simplex sigma points distribution which allows a full parameters space search with minimum amount of sigma points.

2.2.2.4 starSigmaPoints()

```
void SigmaPointsGenerator::starSigmaPoints (
            int nParameters,
            arma::mat * sigma )  [static], [protected]
```

Generates the star sigma points distribution, with the same canonic distribution properties although it increase a sigma point which is the distribution centroid (this implies no perturbation at all).

The documentation for this class was generated from the following files:

- SigmaPointsGenerator.h
- SigmaPointsGenerator.cpp

2.3 StaticROUKF Class Reference

Public Types

- enum **PARAM_TYPE** { **DEFAULT**, **POSITIVE**, **RANGED_LOG_DIST**, **RANGED_NORMAL_DIST** }

Public Member Functions

- StaticROUKF (int nObservations, int nStates, int nParameters, double ∗statesUncertainty, double ∗parametersUncertainty, SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution)
- double executeStep (double ∗Zkhatc, forwardOp A, observationOp H)
- double executeStepParallel (double ∗Zkhatc, forwardOp A, observationOp H, int seed, MPI_Comm local_↩ comm, MPI_Comm masters_comm)
- void **reset** (int nObservations, int nStates, int nParameters, double ∗statesUncertainty, double ∗parameters↩ Uncertainty, SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution)
- void **getParameters** (double ∗∗ThetaC)
- void setParameters (double ∗ThetaC)
- vector< double > **getParametersStd** ()
- void **getError** (double ∗∗err)
- double **getObsError** (int numObservation)
- void toString ()
- int **getObservations** () const
- int **getStates** () const

2.3.1 Constructor & Destructor Documentation

2.3.1.1 StaticROUKF()

```
StaticROUKF::StaticROUKF (
            int nObservations,
            int nStates,
            int nParameters,
            double * observationsUncertainty,
            double * parametersUncertainty,
            SigmaPointsGenerator::SIGMA_DISTRIBUTION sigmaDistribution )
```

Creates the covariance matrixes and sigma points associated with the extended vector X and their uncertainty.

2.3.2 Member Function Documentation

2.3.2.1 executeStep()

```
double StaticROUKF::executeStep (
            double * zkhatc,
            forwardOp A,
            observationOp H )
```

Executes one step of the ROUKF updating X, U, L according to the given measurement zkhatc, assuming an observation variance according to Wi^{-1}.

Forward and observation operators are arguments of the function.

The method return the 2-norm error according to the observations.

2.3.2.2 executeStepParallel()

```
double StaticROUKF::executeStepParallel (
            double * zkhatc,
            forwardOp A,
            observationOp H,
            int sigmaPoint,
            MPI_Comm world_comm,
            MPI_Comm sigmaMasters_comm )
```

MPI Parallel version of ROUKF with one communicator per sigma point sigmaPoint : the sigma point that the current thread must process. world_comm : is the MPI_Comm of the group of all threads that share data in the optimization process. sigmaMasters_comm : is the MPI_Comm with the 0 rank process of the local_comm (each one of them process a different sigma point).

2.3.2.3 setParameters()

```
void StaticROUKF::setParameters (
            double * thetac )
```

Sets the current state of the filter. Once executeStep is executed the state automatically updates.

2.3.2.4 toString()

```
void StaticROUKF::toString ( )
```

Prints the atributes of the ROUKF object.

The documentation for this class was generated from the following files:

- StaticROUKF.h
- StaticROUKF.cpp

Index