MEM Calculators

V00-00-09

Generated by Doxygen 1.8.2

Thu Jan 31 2013 18:46:10

Contents

1 Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

2 Class Index

2.1 Class List

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

3 Namespace Documentation

3.1 MEMNames Namespace Reference

Enumerations

- enum Processes {
  kSMHiggs, **k0hplus**, **k0minus**, **k1plus**,
  k1minus, **k2mplus_gg**, **k2mplus_qqbar**, **kqqZZ**,
  kggZZ, **NUM_PROCESSES** }

Enum type for supported processes in MELA and MEKD packages.

- enum MEMCalcs {
  kAnalytical, **kMEKD**, **kJHUGen**, **kMCFM**,
  kMELA_HCP, **NUM_MEMCALCS** }

  Enum type for supported MEM calculators from MELA and MEKD packages.

3.1.1 Detailed Description

MEMNames namespace provides enum types for names of processes and names of tools/calculators supported by MELA and MEKD packages.

More details can be found at the TWiki: https://twiki.cern.ch/twiki/bin/view/CMS/HZZ4lME

4 Class Documentation

4.1 MEMs Class Reference

Public Types

- enum ERRCodes { **NO_ERR**, **ERR_PROCESS**, **ERR_COMPUTE**, **NUM_ERRORS** }

  enums for supported return values/errors

Public Member Functions

- MEMs (double collisionEnergy=8, string PDFName="", bool debug_=false)
- int computeME (Processes process, MEMCalcs calculator, vector< TLorentzVector > partP, vector< int > partId, double &me2process)
- int computeKD (Processes processA, Processes processB, MEMCalcs calculator, vector< TLorentzVector > partP, vector< int > partId, double &kd, double &me2processA, double &me2processB)
- int computeMEs (vector< TLorentzVector > partP, vector< int > partId)
- int retrieveME (Processes process, MEMCalcs calculator, double &me2process)
- int computeKD (Processes processA, MEMCalcs calculatorA, Processes processB, MEMCalcs calculatorB, double(MEMs::*funcKD)(double, double), double &kd, double &me2processA, double &me2processB)
- int computeKD (Processes processA, MEMCalcs calculatorA, Processes processB, MEMCalcs calculatorB, double(MEMs::*funcKD)(Processes, MEMCalcs, Processes, MEMCalcs), double &kd, double &me2processA, double &me2processB)
- double getMELAWeight ()
- double logRatio (double me2processA, double me2processB)

  Simple KD function: kd = log(me2processA / me2processB).
- double probRatio (Processes processA, MEMCalcs calculatorA, Processes processB, MEMCalcs calculatorB)

  *Case-dependent KD function of a general form: kd = me2processA / (me2processA + c * me2processB).*

Static Public Attributes

- static const bool isProcSupported [NUM_PROCESSES][NUM_MEMCALCS]

  Matrix of supproted processes.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 MEMs::MEMs (double *collisionEnergy* = 8, string *PDFName* = " ", bool *debug_* = false)

Constructor. Can specify the PDF to be use (ony CTEQ6L available at the moment).

Parameters

collisionEnergy	the sqrt(s) value in TeV (DEFAULT = 8).
PDFName	the name of the parton density functions to be used (DEFAULT = "", Optional: "CTEQ6L").

4.1.2 Member Function Documentation

4.1.2.1 int MEMs::computeME (Processes *process*, MEMCalcs *calculator*, vector< TLorentzVector > *partP*, vector< int > *partId*, double & *me2process*)

Compute individual ME for the specified process.

Parameters

in	*process*	names of the process for which the ME should be retrieved.
in	*calculator*	name of the calculator tool to be used.
in	*partP*	the input vector with TLorentzVectors for 4 leptons and 1 photon.
in	*partId*	the input vecor with IDs (PDG) for 4 leptons and 1 photon.
out	*me2process*	retrieved |ME|^2 for the specified process and calculator.

Returns

error code of the computation: 0 = NO_ERR, 1 = ERR_PROCESS, 2 = ERR_COMPUTE

4.1.2.2 int MEMs::computeKD (Processes *processA*, Processes *processB*, MEMCalcs *calculator*, vector< TLorentzVector > *partP*, vector< int > *partId*, double & *kd*, double & *me2processA*, double & *me2processB*)

Compute individual KD and MEs for process A and process B, obtained with the specified calculator tool.

Parameters

in	*processA,processB*	names of the processes A and B for which the KDs and MEs are computed.
in	*calculator*	name of the calculator tool to be used.
in	*partP*	the input vector with TLorentzVectors for 4 leptons and 1 photon.
in	*partId*	the input vecor with IDs (PDG) for 4 leptons and 1 photon.
out	*kd*	computed KD value for discrimination of processes A and B.
out	*me2processA*	computed |ME|^2 for process A.
out	*me2processB*	computed |ME|^2 for process B.

Returns

error code of the computation: 0 = NO_ERR, 1 = ERR_PROCESS, 2 = ERR_COMPUTE

4.1.2.3 int MEMs::computeMEs (vector< TLorentzVector > *partP*, vector< int > *partId*)

Compute MEs for all supported processes.

Individual MEs and KDs can be retrieved using retrieveME(Processes,MEMCalcs,double&) and computeKD(Processes,MEMCalcs,Processes,MEMCalcs,double(*)(double,double),double&,double&,double&).

Parameters

in	*partP*	the input vector with TLorentzVectors for 4 leptons and 1 photon.
in	*partId*	the input vecor with IDs (PDG) for 4 leptons and 1 photon.

Returns

error code of the computation: 0 = NO_ERR, 2 = ERR_COMPUTE

4.1.2.4 int MEMs::retrieveME (Processes *process*, MEMCalcs *calculator*, double & *me2process*)

Retrieve ME for specified process and specified calculator tool.

Method should be called only after running computeMEs(vector<TLorentzVector> partP,vector<int> partId).

Parameters

in	*process*	names of the process for which the ME should be retrieved.
in	*calculator*	name of the calculator tool to be used.
out	*me2process*	retrieved |ME|^2 for the specified process and calculator.

Returns

error codes: 0 = NO_ERR, 1 = ERR_PROCESS

4.1.2.5 int MEMs::computeKD (Processes *processA*, MEMCalcs *calculatorA*, Processes *processB*, MEMCalcs *calculatorB*, double(MEMs::*)(double, double) *funcKD*, double & *kd*, double & *me2processA*, double & *me2processB*)

Compute KD and retrieve MEs for process A and process B, obtained with the specified calculator tool. The KD is computed using KD function specified by the user as kd = funcKD(me2processA, me2processB).

Method should be called only after running computeMEs(vector<TLorentzVector> partP,vector<int> partId).

Parameters

in	*processA,processB*	names of the processes for which the KD and MEs are computed.
in	*calculatorA,calculatorB*	names of the calculator tools to be used.
in	*funcKD*	name of the method to be used for KD computation.
out	*kd*	computed KD value for discrimination of processes A and B.
out	*me2processA*	computed |ME|^2 for process A.
out	*me2processB*	computed |ME|^2 for process B.

Returns

error code of the computation: 0 = NO_ERR, 1 = ERR_PROCESS

4.1.2.6 int MEMs::computeKD (Processes *processA*, MEMCalcs *calculatorA*, Processes *processB*, MEMCalcs *calculatorB*, double(MEMs::*)(Processes, MEMCalcs, Processes, MEMCalcs) *funcKD*, double & *kd*, double & *me2processA*, double & *me2processB*)

Compute KD and retrieve MEs for process A and process B, obtained with the specified calculator tool. The KD is computed using KD function specified by the user which has defferent implementations for different combinations of processes and calculator tools. Functions is of the form kd = funcKD(processA, calculatorA, processB, calculatorB).

Method should be called only after running computeMEs(vector<TLorentzVector> partP,vector<int> partId).

Parameters

in	*processA,processB*	names of the processes for which the KD and MEs are computed.
in	*calculatorA,calculatorB*	names of the calculator tools to be used.
in	*funcKD*	name of the method to be used for KD computation.
out	*kd*	computed KD value for discrimination of processes A and B.
out	*me2processA*	computed |ME|^2 for process A.
out	*me2processB*	computed |ME|^2 for process B.

Returns

error code of the computation: 0 = NO_ERR, 1 = ERR_PROCESS

4.1.2.7 double MEMs::getMELAWeight ( ) [inline]

Retrieve the interference reweighting factor for the given event, computed using the JHUGen.

Method should be called only after running computeMEs(vector<TLorentzVector> partP,vector<int> partId).

Returns

interference reweighting factor for the given event.

4.1.3 Member Data Documentation

4.1.3.1 const bool MEMs::isProcSupported [static]

Initial value:

```
= {

  {1,           1,          1,          1,          1},
  {1,           1,          1,          0,          0},
  {1,           1,          1,          0,          0},
  {1,           1,          1,          0,          0},
  {1,           1,          1,          0,          0},
  {1,           1,          1,          0,          0},
  {1,           1,          1,          0,          0},
  {1,           1,          0,          1,          1},
  {0,           0,          0,          1,          0}}
```

Matrix of supproted processes.

Matrix of supproted processes - initialisation (to be updated)

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

- MEMCalculators.h

Index