LPAIR++

0.1

Generated by Doxygen 1.8.3.1

Tue Jul 16 2013 15:15:00

Contents

1 Todo List

Global GamGam::GamGam (const unsigned int, double, double, int, double x_[])

Figure out how this nOpt_ parameter is affecting the final cross-section computation and events generation

What are these w12, w31, w52 parameters introduced in the GAMGAM subroutine ? And why are they set to 0. ?

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 Data Structure Documentation

3.1 Cuts Class Reference

List of kinematic cuts to apply on the central and outgoing phase space.

Public Member Functions

- Cuts ()
- ~Cuts ()

Data Fields

- double emax
  Maximal energy of the central two-photons system.
- double emin
  Minimal energy of the central two-photons system.
- int mode
  Sets of cuts to apply on the final phase space.
- double mxmax
- double mxmin
- double ptmax
  Maximal transverse momentum of the single outgoing leptons.
- double ptmin
  Minimal transverse momentum of the single outgoing leptons.
- double thetamax
  Maximal polar ($\theta_{\max}$) angle of the outgoing leptons, expressed in degrees.
- double thetamin
  Minimal polar ($\theta_{\min}$) angle of the outgoing leptons, expressed in degrees.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Cuts::Cuts ( )

3.1.1.2 Cuts::~Cuts ( )

3.1.2 Field Documentation

3.1.2.1 double Cuts::emax

3.1.2.2 double Cuts::emin

3.1.2.3 int Cuts::mode

3.1.2.4 double Cuts::mxmax

3.1.2.5 double Cuts::mxmin

3.1.2.6 double Cuts::ptmax

3.1.2.7 double Cuts::ptmin

3.1.2.8 double Cuts::thetamax

3.1.2.9 double Cuts::thetamin

3.2 GamGam Class Reference

Computes the matrix element for a $\gamma\gamma \to \ell^+\ell^-$ process.

Public Member Functions

- GamGam (const unsigned int, double, double, int, double x_[])
  Class constructor.
- ~GamGam ()
- void ComputeSqS ()
  Computes $\sqrt{s}$ for the system.
- double ComputeXsec (int nm_=1)
  Computes the process' cross section.
- void FillKinematics ()
- Particle GetParticle (int)
  Get a particle given its role in the process.
- bool IsKinematicsDefined ()
  Is the system's kinematics well defined?
- void SetCuts (Cuts)
  Sets the list of kinematic cuts to apply on the outgoing particles' final state.
- bool SetIncomingKinematics (int, double[], int)
  Sets the momentum and PDG id for the incoming particles.
- bool SetOutgoingParticles (int, int)
  Sets the PDG id for the outgoing particles.
- void SetWRange (double, double)
  Sets the energy range available for the phase space integration.
- void StoreEvent (std::ofstream *, double)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 GamGam::GamGam (const unsigned int *ndim_*, double *q2min_*, double *q2max_*, int *nOpt_*, double *x_[]*)

Sets the mandatory parameters used in the methods computing the kinematics and the cross-section of this phase space point.

Parameters

*ndim_*	The number of dimensions of the point in the phase space
*q2min_*	The minimal value of Q^2
*q2max_*	The maximal value of Q^2
*nOpt_*	Optimisation???
x_[]	The ndim_-dimensional point in the phase space on which the kinematics and the cross-section are computed

Todo Figure out how this nOpt_ parameter is affecting the final cross-section computation and events generation

What are these w12, w31, w52 parameters introduced in the GAMGAM subroutine ? And why are they set to 0. ?

3.2.1.2 GamGam::~GamGam ( )

3.2.2 Member Function Documentation

3.2.2.1 void GamGam::ComputeSqS ( )

Computes the centre of mass energy for the system, according to the incoming particles' kinematics

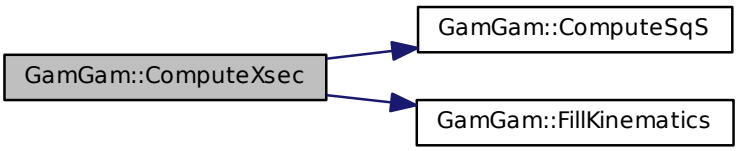
3.2.2.2 double GamGam::ComputeXsec (int *nm_* = 1)

Computes the cross-section for the $\gamma\gamma \to \ell^+\ell^-$ process with the given kinematics

Returns

$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{x}}(\gamma\gamma \to \ell^+\ell^-)$, the differential cross-section for the given point in the phase space.

Here is the call graph for this function:



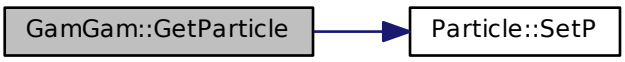
3.2.2.3 void GamGam::FillKinematics ( )

3.2.2.4 Particle GamGam::GetParticle (int *role_*)

Parameters

*role_*	An integer denoting the particle's role in the selected production process

Here is the call graph for this function:



3.2.2.5 bool GamGam::IsKinematicsDefined ( ) [inline]

Is the system's kinematics well defined and compatible with the process ? This check is mandatory to perform the (_ndim)-dimensional point's cross-section computation.

Returns

A boolean stating if the input kinematics and the final states are well defined

3.2.2.6 void GamGam::SetCuts (Cuts *cuts_*)

Parameters

*cuts_*	The Cuts object containing the kinematic parameters

3.2.2.7 bool GamGam::SetIncomingKinematics (int , double *[]*, int)

Specifies the incoming particles' kinematics as well as their properties (role in the process and PDG Id)

Parameters

*part_*	Role of the particle in the process
momentum_[]	3-momentum of the particle
*pdgId_*	Particle ID according to the PDG convention

Returns

True if the kinematics was correctly set for the given particle role

3.2.2.8 bool GamGam::SetOutgoingParticles (int *part_*, int *pdgId_*)

Parameters

*part_*	Role of the particle in the process
*pdgId_*	Particle ID according to the PDG convention

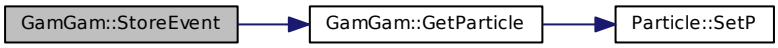
3.2.2.9 void GamGam::SetWRange (double *wmin_*, double *wmax_*)

Parameters

*wmin_*	The minimal s on which the cross section is integrated
*wmax_*	The maximal s on which the cross section is integrated. If negative, the maximal energy available to the system (hence, $s = (\sqrt{s})^2$) is provided.

3.2.2.10 void GamGam::StoreEvent (std::ofstream * *file_*, double *weight_*)

Here is the call graph for this function:



3.3 InputParameters Class Reference

List of input parameters used to start and run the simulation job.

Public Member Functions

- InputParameters ()
- ~InputParameters ()
- bool ReadConfigFile (std::string)

  Reads content from config file to load the variables.
- bool StoreConfigFile (std::string)

  Stores the full run configuration to an external config file.

Data Fields

- bool debug

  Do we need control plots all along the process?
- std::ofstream * file

  The file in which to store the events generation's output.
- bool generation

  Are we generating events ? (true) or are we only computing the cross-section ? (false)
- double in1p

  First incoming particle's momentum (in GeV/c)
- double in2p

  Second incoming particle's momentum (in GeV/c)
- int itvg

  Number of Vegas integrations.
- double maxenergy
- double maxmx
- double maxpt

  Maximal transverse momentum of the outgoing leptons.
- double maxtheta
- int mcut

  Set of cuts to apply on the outgoing leptons.
- double minenergy
- double minmx
- double minpt

  Minimal transverse momentum of the outgoing leptons.
- double mintheta
- int ncvg

- int ngen
- int p1mod

  First particle's mode.
- int p2mod

  Second particle's mode.
- int pair

  PDG id of the outgoing leptons.
- Gnuplot * plot [MAX_HISTOS]

  Control plots objects.
- bool store

3.3.1 Detailed Description

Note

The default parameters are derived from GMUINI in LPAIR

3.3.2 Constructor & Destructor Documentation

3.3.2.1 InputParameters::InputParameters ( )

3.3.2.2 InputParameters::~InputParameters ( )

3.3.3 Member Function Documentation

3.3.3.1 bool InputParameters::ReadConfigFile (std::string *inFile_*)

Parameters

*inFile_*	Name of the configuration file to load

3.3.3.2 bool InputParameters::StoreConfigFile (std::string *outFile_*)

Parameters

*outFile_*	Name of the configuration file to create

3.3.4 Field Documentation

3.3.4.1 bool InputParameters::debug

Enables or disables the production of control plots for several kinematic quantities in this process

3.3.4.2 std::ofstream* InputParameters::file

3.3.4.3 bool InputParameters::generation

3.3.4.4 double InputParameters::in1p

3.3.4.5 double InputParameters::in2p

3.3.4.6 int InputParameters::itvg

3.3.4.7 double InputParameters::maxenergy

3.3.4.8 double InputParameters::maxmx

3.3.4.9 double InputParameters::maxpt

3.3.4.10 double InputParameters::maxtheta

3.3.4.11 int InputParameters::mcut

Set of cuts to apply on the outgoing leptons in order to restrain the available kinematic phase space :

- 0 - No cuts at all (for the total cross section)
- 1 - Vermaserens' hypothetical detector cuts : for both leptons,
  - $\frac{|p_z|}{|\mathbf{p}|} \leq 0.75$ and $p_T \geq 1$ GeV, or
  - $0.75 < \frac{|p_z|}{|\mathbf{p}|} \leq 0.95$ and $p_z > 1$ GeV,
- 2 - Cuts according to the provided parameters

3.3.4.12 double InputParameters::minenergy

3.3.4.13 double InputParameters::minmx

3.3.4.14 double InputParameters::minpt

3.3.4.15 double InputParameters::mintheta

3.3.4.16 int InputParameters::ncvg

3.3.4.17 int InputParameters::ngen

3.3.4.18 int InputParameters::p1mod

The first incoming particle type and kind of interaction :

- 1 - electron,
- 2 - proton elastic,
- 3 - proton inelastic without parton treatment,
- 4 - proton inelastic in parton model

Note

Was named PMOD in ILPAIR

3.3.4.19 int InputParameters::p2mod

Note

Was named EMOD in ILPAIR

3.3.4.20 int InputParameters::pair

The particle code of produced leptons :

- 11 - for e^+e^- pairs
- 13 - for $\mu^+\mu^-$ pairs
- 15 - for $\tau^+\tau^-$ pairs

3.3.4.21 Gnuplot* InputParameters::plot[MAX_HISTOS]

List of Gnuplot objects which can be used to produce control plots all along the cross-section determination and events generation process

Note

Maximum number of these can be raised in the utils.h file, but pay attention to the memory load since these Gnuplot objects are still under development!

3.3.4.22 bool InputParameters::store

3.4 MCGen Class Reference

Core of the Monte-Carlo generator ; Computes the cross section for any value of the input parameters by calling Vegas on GamGam objects.

Public Member Functions

- MCGen (InputParameters)
  Class constructor.
- ~MCGen ()
- void AnalyzePhaseSpace (const std::string)
- void ComputeXsection (double *, double *)
- InputParameters GetInputParameters ()
  Returns the set of parameters used to setup the phase space to integrate.
- void LaunchGen (const unsigned int)
- void Test ()

3.4.1 Detailed Description

This object represents the core of this Monte Carlo generator, with its allowance to generate the events (using the embedded Vegas object) and to study the phase space in term of the variation of resulting cross section while scanning the various parameters (point **x** in the DIM-dimensional phase space).

The phase space is constrained using the InputParameters object given as an argument to the constructor, and the differential cross-sections for each value of the array **x** are computed in the f-function defined outside (but populated inside) this object.

This f-function embeds a GamGam object which defines all the methods to obtain this differential cross-section as well as the in- and outgoing kinematics associated to each particle.

Author

Laurent Forthomme laurent.forthomme@uclouvain.be

Date

February 2013

3.4.2 Constructor & Destructor Documentation

3.4.2.1 MCGen::MCGen (InputParameters *ip_*)

Sets the number of dimensions on which to perform the integration, according to the set of input parameters given as an argument and propagated to the whole object

Parameters

*ip_*	List of input parameters defining the phase space on which to perform the integration

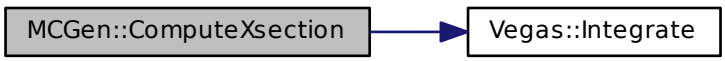
3.4.2.2 MCGen::~MCGen ( )

3.4.3 Member Function Documentation

3.4.3.1 void MCGen::AnalyzePhaseSpace (const std::string *outputFile_*)

3.4.3.2 void MCGen::ComputeXsection (double * *xsec_*, double * *err_*)

Here is the call graph for this function:



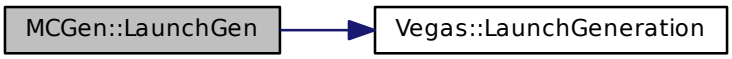
3.4.3.3 InputParameters MCGen::GetInputParameters ( ) [inline]

Returns

The InputParameter object embedded in this class

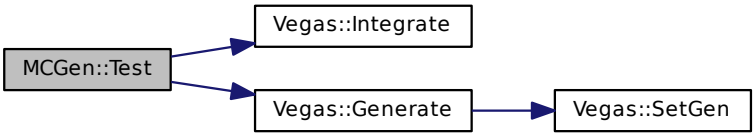
3.4.3.4 void MCGen::LaunchGen (const unsigned int *count_*)

Here is the call graph for this function:



3.4.3.5 void MCGen::Test ( )

Here is the call graph for this function:



3.5 Particle Class Reference

Kinematics of one particle.

Public Member Functions

- Particle ()
- ~Particle ()
- std::string GetLHEline ()
- void SetP (double, double, double)

  Sets the momentum.

Data Fields

- double e

  Energy in GeV.
- double m

  Mass in GeV/c ^{2}.
- int pdgId

  Particle Data Group integer identifier.
- double pt

  Transverse momentum.
- double px

  Momentum along the x*-axis in GeV/c.*
- double py

  Momentum along the y*-axis in GeV/c.*
- double pz

  Momentum along the z*-axis in GeV/c.*
- int role

  Role in the considered process.

3.5.1 Detailed Description

Kinematic information for one particle

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Particle::Particle ( )

3.5.2.2 Particle::~Particle ( )

3.5.3 Member Function Documentation

3.5.3.1 std::string Particle::GetLHEline ( )

Returns a string containing all the particle's kinematics as expressed in the Les Houches format

Returns

The LHE line

3.5.3.2 void Particle::SetP (double *px_*, double *py_*, double *pz_*)

Parameters

*px_*	Momentum along the x-axis
*py_*	Momentum along the y-axis
*pz_*	Momentum along the z-axis

3.5.4 Field Documentation

3.5.4.1 double Particle::e

3.5.4.2 double Particle::m

3.5.4.3 int Particle::pdgId

3.5.4.4 double Particle::pt

3.5.4.5 double Particle::px

3.5.4.6 double Particle::py

3.5.4.7 double Particle::pz

3.5.4.8 int Particle::role

3.6 Vegas Class Reference

Vegas Monte-Carlo integrator instance.

Public Member Functions

- Vegas (int, double f_(double *, size_t, void *), InputParameters *inParam_)
- ~Vegas ()

  Class destructor.
- int Generate (int)
- int Integrate (double *, double *)

  Launches the integration of the provided function.
- int LaunchGeneration (int)

  Launches the generation of events.
- void SetGen ()

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Vegas::Vegas (int *dim_*, double *f_double **, *size_t, void **, **InputParameters** * *inParam_*)

Constructs the class by booking the memory and structures for the GSL Vegas integrator. This code from the GNU scientific library is based on the Vegas Monte Carlo integration algorithm developed by P. Lepage. [1]

Parameters

*dim_*	The number of dimensions on which the function will be integrated
*f_*	The function one is required to integrate
*inParam_*	A list of parameters to define the phase space on which this integration is performed (embedded in an InputParameters object)

3.6.1.2 Vegas::~Vegas ( )

3.6.2 Member Function Documentation

3.6.2.1 int Vegas::Generate (int *nEvt_*)

Here is the call graph for this function:

3.6.2.2 int Vegas::Integrate (double * *result_*, double * *abserr_*)

Launches the Vegas integration of the provided function with the provided input parameters.

Parameters

*result_*	The cross section as integrated by Vegas for the given phase space restrictions
*abserr_*	The error associated to the computed cross section

3.6.2.3 int Vegas::LaunchGeneration (int)

Launches the Vegas generation of events according to the provided input parameters.

Parameters

*nEvts_*	The number of events to generate

3.6.2.4 void Vegas::SetGen ( )

XXXXXX.....

References

[1] G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192 – 203, 1978.

Index