LPAIR++

0.1

Generated by Doxygen 1.8.3.1

Mon May 6 2013 20:01:31

Contents

1 Todo List

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

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

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 ptmax
  Maximal transverse momentum of the single outgoing leptons.

- double ptmin
  Minimal transverse momentum of the single outgoing leptons.
- double thetamax
  Maximal polar (θ) angle of the outgoing leptons.
- double thetamin
  Minimal polar (θ) angle of the outgoing leptons.

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::ptmax

3.1.2.5 double Cuts::ptmin

3.1.2.6 double Cuts::thetamax

3.1.2.7 double Cuts::thetamin

3.2 GamGam Class Reference

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

Public Member Functions

- GamGam (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?
- bool Orient ()
  Energies/momenta computation for the various particles, in the CM system.
- double PeriPP (int, int)
  Computes the matrix element squared for the requested process.
- bool Pickin ()
- 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.

3.2.1 Constructor & Destructor Documentation

3.2.1.1 GamGam::GamGam (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

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

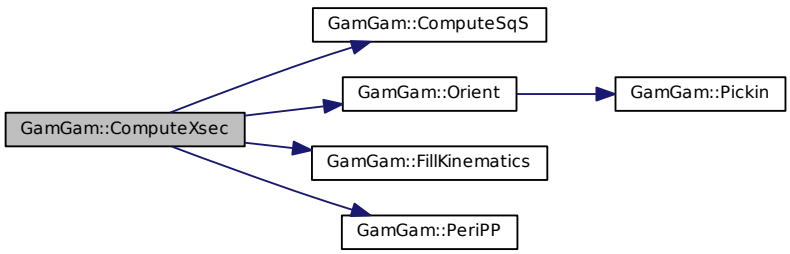
3.2.2.2 double GamGam::ComputeXsec (int *nm_* = 1)

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:



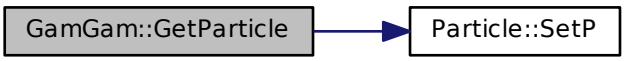
3.2.2.3 void GamGam::FillKinematics ( )

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 bool GamGam::Orient ( )

Calculates energies and momenta of the 1st, 2nd (resp. the "proton-like" and the "electron-like" incoming particles), 3rd (the "proton-like" outgoing particle), 4th (the two-photons central system) and 5th (the "electron-like" outgoing particle) particles in the overall centre of mass frame.

Here is the call graph for this function:

3.2.2.7 double GamGam::PeriPP (int *nup_*, int *ndown_*)

Contains the expression of the matrix element squared for the process under considerations. It returns the value of the convolution of the form factor or structure functions with the central two-photons matrix element squared.

Returns

The full matrix element for the two-photon production of a pair of spin $-\frac{1}{2}-$point particles

3.2.2.8 bool GamGam::Pickin ( )

Describes the kinematics of the process $p_1 + p_2 \to p_3 + p_4 + p_5$ in terms of Lorentz-invariant variables. These variables (along with others) will then be feeded into the PeriPP method (thus are essential for the evaluation of the full matrix element).

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

Parameters

*cuts_*	The Cuts object containing the kinematic parameters

3.2.2.10 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

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

Parameters

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

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

Parameters

*wmin_*	The minimal s on which the cross section is integrated
*wmin_*	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.

3.3 Gnuplot Class Reference

Plotting utility used in control plots generation.

Public Member Functions

- Gnuplot (std::string outFile_="")
- ~Gnuplot ()
- int DrawHistogram ()
- int Fill (double)

  Add an entry to the histogram.
- int Fill (double, double)

  Add an entry to the histogram.
- int Fill2 (double, double)
- int Fill2 (double, double, double)
- void operator<< (const std::string &command)

  Feeds a command line to the Gnuplot interpreter.
- void SetGrid (bool grid_=true)

  Toggles the logarithmic scale for the y-axis.
- void SetHistogram (int, double, double, std::string name_="")

- void SetLogy (bool logy_=true)
  Toggles the grid for both the axes.
- void SetName (std::string)
  Sets the name for the graph.
- void SetOutputFile (std::string)
- void SetTitle (std::string)
  Sets the title for the graph.
- void SetXAxisTitle (std::string)
  Sets the caption for the x-axis.
- void SetYAxisTitle (std::string)
  Sets the caption for the y-axis.

Protected Attributes

- FILE * _pipe
  The pipe used to feed the Gnuplot interpreter.

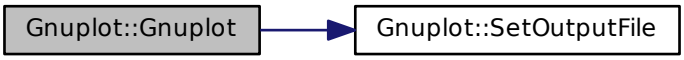
3.3.1 Detailed Description

This object allows to invoke gnuplot, the portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Gnuplot::Gnuplot (std::string *outFile_* = "")

Here is the call graph for this function:



3.3.2.2 Gnuplot::~Gnuplot ( )

3.3.3 Member Function Documentation

3.3.3.1 int Gnuplot::DrawHistogram ( )

Here is the call graph for this function:

3.3.3.2 int Gnuplot::Fill (double *value_*)

Adds an entry to the histogram in case the current object is set as a GP_HISTOGRAM-type plotter

Parameters

*value_*	The value to insert into the histogram

Returns

An integer less than or equal 0 if an error has been observed, or

- 1 if the value was within the histogram's range
- 2 if the value was set in the underflow bin
- 3 if the value was set in the overflow bin

3.3.3.3 int Gnuplot::Fill (double *value_*, double *weight_*)

Adds an entry to the histogram in case the current object is set as a GP_HISTOGRAM-type plotter

Parameters

*value_*	The value to insert into the histogram
*weight_*	The weight of this entry in the histogram

Returns

An integer less than or equal 0 if an error has been observed, or

- 1 if the value was within the histogram's range
- 2 if the value was set in the underflow bin
- 3 if the value was set in the overflow bin

3.3.3.4 int Gnuplot::Fill2 (double , double)

3.3.3.5 int Gnuplot::Fill2 (double , double , double)

3.3.3.6 void Gnuplot::operator<< (const std::string & *command*)

Simplest way to use this object if you know how to plot things with gnuplot : just create the interpreter, and then feed your commands to generate the graph.

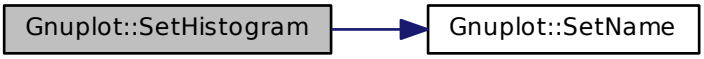
Parameters

&command	The Gnuplot-formatted command line to feed

3.3.3.7 void Gnuplot::SetGrid (bool *grid_* = `true`)

3.3.3.8 void Gnuplot::SetHistogram (int *num_*, double *low_*, double *high_*, std::string *name_* = "")

Here is the call graph for this function:



3.3.3.9 void Gnuplot::SetLogy (bool *logy_* = `true`)

3.3.3.10 void Gnuplot::SetName (std::string *name_*)

3.3.3.11 void Gnuplot::SetOutputFile (std::string *outFile_*)

Sets the file on which the graph has to be produced

3.3.3.12 void Gnuplot::SetTitle (std::string *title_*)

3.3.3.13 void Gnuplot::SetXAxisTitle (std::string *title_*)

3.3.3.14 void Gnuplot::SetYAxisTitle (std::string *title_*)

3.3.4 Field Documentation

3.3.4.1 FILE* Gnuplot::_pipe `[protected]`

3.4 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 maxpt

  Maximal transverse momentum of the outgoing leptons.
- int mcut

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

  Minimal transverse momentum of the outgoing leptons.
- 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.

3.4.1 Constructor & Destructor Documentation

3.4.1.1 InputParameters::InputParameters ( )

3.4.1.2 InputParameters::~InputParameters ( )

3.4.2 Member Function Documentation

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

Parameters

*inFile_*	Name of the configuration file to load

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

Parameters

*outFile_*	Name of the configuration file to create

3.4.3 Field Documentation

3.4.3.1 bool InputParameters::debug

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

3.4.3.2 std::ofstream* InputParameters::file

3.4.3.3 bool InputParameters::generation

3.4.3.4 double InputParameters::in1p

3.4.3.5 double InputParameters::in2p

3.4.3.6 int InputParameters::itvg

3.4.3.7 double InputParameters::maxpt

3.4.3.8 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

3.4.3.9 double InputParameters::minpt

3.4.3.10 int InputParameters::ncvg

3.4.3.11 int InputParameters::ngen

3.4.3.12 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

3.4.3.13 int InputParameters::p2mod

Note

Was named EMOD in ILPAIR

3.4.3.14 int InputParameters::pair

The particle code of produced leptons :

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

3.4.3.15 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!

3.5 MCGen Class Reference

Core Monte-Carlo generator.

Public Member Functions

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

3.5.1 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

3.5.2 Constructor & Destructor Documentation

3.5.2.1 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

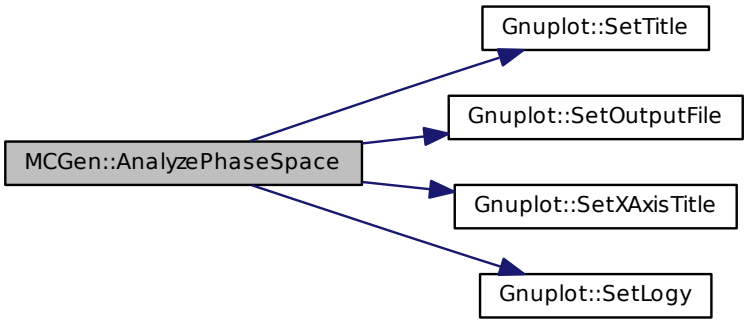
*ip_*	List of input parameters defining the phase space on which to perform the integration

3.5.2.2 MCGen::~MCGen ( )

3.5.3 Member Function Documentation

3.5.3.1 void MCGen::AnalyzePhaseSpace (std::string *outputFile_*)

Here is the call graph for this function:



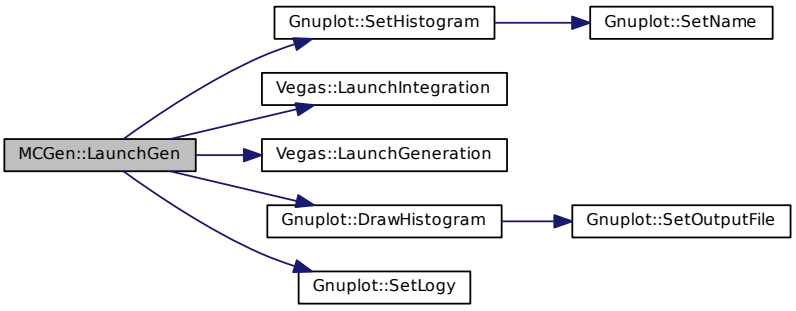
3.5.3.2 InputParameters MCGen::GetInputParameters ( ) [inline]

Returns

The InputParameter object embedded in this class

3.5.3.3 void MCGen::LaunchGen (int *count_*)

Here is the call graph for this function:



3.6 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.
- 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.

3.6.1 Detailed Description

Kinematic information for one particle

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Particle::Particle ( )

3.6.2.2 Particle::~Particle ( )

3.6.3 Member Function Documentation

3.6.3.1 std::string Particle::GetLHEline ( )

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

Returns

The LHE line

3.6.3.2 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

3.6.4 Field Documentation

3.6.4.1 double Particle::e

3.6.4.2 double Particle::m

3.6.4.3 int Particle::pdgId

3.6.4.4 double Particle::pt

3.6.4.5 double Particle::px

3.6.4.6 double Particle::py

3.6.4.7 double Particle::pz

3.6.4.8 int Particle::role

3.7 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 LaunchGeneration (int)

  Launches the generation of events.
- int LaunchIntegration ()

  Launches the integration of the provided function.

3.7.1 Constructor & Destructor Documentation

3.7.1.1 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)

3.7.1.2 Vegas::~Vegas ( )

3.7.2 Member Function Documentation

3.7.2.1 int Vegas::LaunchGeneration (int *nEvts_*)

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

Parameters

*nEvts_*	The number of events to generate

3.7.2.2 int Vegas::LaunchIntegration ( )

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

References

[1] G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192 – 203, 1978. 15

Index