LPAIR++

0.2

Generated by Doxygen 1.8.6

Tue Jan 14 2014 18:09:46

Contents

1 Principles

This Monte Carlo generator, based on the LPAIR code developed in the early 1990s by J. Vermaseren *et al*[3], allows to compute the cross-section and to generate events for the $\gamma\gamma \to \ell^+\ell^-$ process in high energy physics.

The main operation is the integration of the matrix element (given as a GamGam object, subset of a Process object) performed by *Vegas*, an importance sampling algorithm written in 1972 by G. P. Lepage[2].

2 Todo List

Global GamGam::ComputeWeight (int nm_=1)

Find out what this *nm_* parameter does...

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

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

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

4 Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

5 Data Structure Documentation

5.1 Event Class Reference

Kinematic information on the particles in the event.

Public Member Functions

- int AddParticle (Particle $*$part_)

  Add a particle to the event.
- void Dump ()
- Particle $*$ GetById (int id_)

  Gets one particle by its unique identifier in the event.
- Particle $*$ GetByRole (int role_)

  Gets one particle by its role in the event.
- std::vector$<$ Particle $*$ $>$ GetParticles ()

Gets a vector of particles in the event.
- int NumParticles ()

  Number of particles in the event.
- void Store (std::ofstream ∗, double weight_=1.)
- void StoreLHERecord (std::ofstream ∗of_, const double weight_=1.)

  Stores the LHE block for this event.

5.1.1 Detailed Description

Class containing all the information on the in- and outgoing particles' kinematics

5.1.2 Member Function Documentation

5.1.2.1 int Event::AddParticle (**Particle** ∗ part_)

Sets the information on one particle in the process

Parameters

*part_*	The Particle object to insert or modify in the event

Returns

- 1 if a new Particle object has been inserted in the event
- 0 if an existing Particle object has been modified
- -1 if the requested role to edit is undefined or incorrect

5.1.2.2 void Event::Dump ( )

Dumps all the known information on every Particle object contained in this Event container in the output stream

5.1.2.3 **Particle**∗ Event::GetById (int id_)

Returns the pointer to the Particle object corresponding to a unique identifier in the event

Parameters

*id_*	The unique identifier to this particle in the event

Returns

A pointer to the requested Particle object

5.1.2.4 **Particle**∗ Event::GetByRole (int role_)

Returns the pointer to the Particle object corresponding to a certain role in the process kinematics

Parameters

*role_*	The role the particle has to play in the process

Returns

A pointer to the requested Particle object

5.1.2.5 std::vector<**Particle***> Event::GetParticles ( )

Returns

A vector containing all the pointers to the Particle objects contained in the event

5.1.2.6 int Event::NumParticles ( ) [inline]

Returns

The number of particles in the event, as an integer

5.1.2.7 void Event::Store (std::ofstream * , double weight_ = 1.)

Stores in a file (raw format) all the kinematics on the outgoing leptons

Parameters

*weight_*	The weight of the event

5.1.2.8 void Event::StoreLHERecord (std::ofstream * of_, const double weight_ = 1.)

Stores in a LHE format (a XML-style) all the information on the particles composing this event

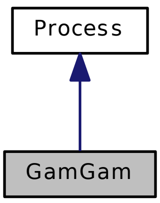
Parameters

*of_*	The file stream on which the event record has to be saved
*weight_*	The weight of the event

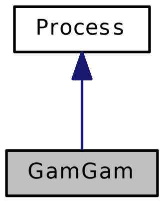
5.2 GamGam Class Reference

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

Inheritance diagram for GamGam:



Collaboration diagram for GamGam:



Public Member Functions

- GamGam (const unsigned int ndim_, int nOpt_, double x_[])

    Class constructor.
- void ComputeCMenergy ()

    Computes $\sqrt{s}$ *for the system.*
- double ComputeMX (double x_, double outmass_, double *dw_)

    Computes the ougoing proton remnant mass.
- double ComputeWeight ()

    Returns the weight for this point in the phase-space.
- double ComputeWeight (int nm_=1)

    Computes the process' weight for the given point.
- void FillKinematics (bool symmetrise_=false)

    Fills the Event object with the particles' kinematics.
- double **GetD3** ()
- Event * GetEvent ()

    Returns the event content (list of particles with an assigned role)
- double **GetS1** ()
- double **GetS2** ()
- double GetT1 ()
- void GetT1extrema (double &t1min_, double &t1max_)
- double GetT2 ()
- void GetT2extrema (double &t2min_, double &t2max_)
- double **GetU1** ()
- double **GetU2** ()
- double **GetV1** ()
- double **GetV2** ()
- bool IsKinematicsDefined ()

    Is the system's kinematics well defined?
- void PrepareHadronisation (Particle *part_)
- bool SetIncomingKinematics (Particle ip1_, Particle ip2_)

    Sets the momentum and PDG id for the incoming particles.
- void SetKinematics (GamGamKinematics cuts_)

    Sets the list of kinematic cuts to apply on the outgoing particles' final state.
- bool SetOutgoingParticles (int part_, int pdgId_)

    Sets the PDG id for the outgoing particles.
- void **StoreEvent** (std::ofstream *, double)

5.2.1 Detailed Description

Full class of methods and objects to compute the full analytic matrix element [3] for the $\gamma\gamma \to \ell^+\ell^-$ process according to a set of kinematic constraints provided for the incoming and outgoing particles (the GamGamKinematics object).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 GamGam::GamGam (const unsigned int ndim_, 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
*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

5.2.3 Member Function Documentation

5.2.3.1 void GamGam::ComputeCMenergy ( )

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

5.2.3.2 double GamGam::ComputeMX (double x_, double outmass_, double * dw_)

Computes the mass of the outgoing proton remnant if any

Parameters

*x_*	A random number (between 0 and 1)
*outmass_*	The maximal outgoing particles' invariant mass
*dw_*	The size of the integration bin

Returns

The mass of the outgoing proton remnant

5.2.3.3 double GamGam::ComputeWeight (int nm_ = 1)

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

Parameters

*nm_*	???

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.

Todo Find out what this *nm_* parameter does...

5.2.3.4 void GamGam::FillKinematics (bool symmetrise_ = false)

Fills the private Event object with all the Particle object contained in this event. The particle roles in this process are defined as following :

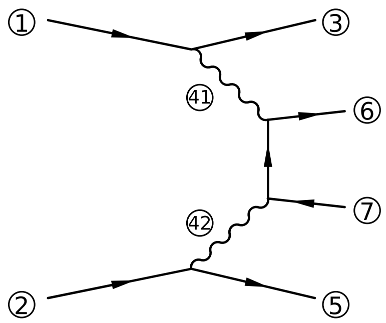


Figure 1: Detailed particle roles in the two-photon process as defined by the *GamGam* object. The incoming protons/electrons are denoted by a role 1, and 2, as the outgoing protons/protons remnants/electrons carry the indices 3 and 5. The two outgoing leptons have the roles 6 and 7, while the lepton/antilepton distinction is done randomly (thus, the arrow convention is irrelevant here).

5.2.3.5 **Event**∗ GamGam::GetEvent ( ) [inline]

Returns the complete list of Particle with their role in the process for the point considered in the phase space as an Event object.

Returns

The Event object containing all the generated Particle objects

5.2.3.6 double GamGam::GetT1 ( ) [inline]

Returns the value for the first photon virtuality

Returns

t_1, the first photon virtuality

5.2.3.7 void GamGam::GetT1extrema (double & t1min_, double & t1max_) [inline]

Returns the two limit values for the first photon virtuality

Parameters

*t1min_*	The minimal value for t_1
*t1max_*	The maximal value for t_1

5.2.3.8 double GamGam::GetT2 ( ) [inline]

Returns the value for the second photon virtuality

Returns

t_2, the second photon virtuality

5.2.3.9 void GamGam::GetT2extrema (double & t2min_, double & t2max_) [inline]

Returns the two limit values for the second photon virtuality

Parameters

*t2min_*	The minimal value for t_2
*t2max_*	The maximal value for t_2

5.2.3.10 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

5.2.3.11 void GamGam::PrepareHadronisation (**Particle** * part_)

Sets all the kinematic variables for the outgoing proton remnants in order to be able to hadronise them afterwards

Parameters

*part_*	Particle to "prepare" for the hadronisation to be performed

5.2.3.12 bool GamGam::SetIncomingKinematics (**Particle** ip1_, **Particle** ip2_)

Specifies the incoming particles' kinematics as well as their properties using two Particle objects.

Parameters

*ip1_*	Information on the first incoming particle
*ip2_*	Information on the second incoming particle

5.2.3.13 void GamGam::SetKinematics (**GamGamKinematics** cuts_)

Parameters

*cuts_*	The Cuts object containing the kinematic parameters

5.2.3.14 bool GamGam::SetOutgoingParticles (int part_, int pdgId_)

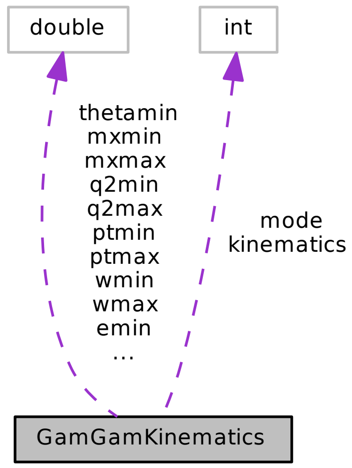
Parameters

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

5.3 GamGamKinematics Class Reference

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

Collaboration diagram for GamGamKinematics:



Public Member Functions

- void **Dump** ()

Data Fields

- double emax

  Maximal energy of the central two-photons system.
- double emin

  Minimal energy of the central two-photons system.
- int kinematics

  Type of kinematics to consider for the phase space.
- int mode

  Sets of cuts to apply on the final phase space.
- double mxmax

  Maximal mass (in GeV/c ^{2}) of the outgoing proton remnant(s)
- double mxmin

  Minimal mass (in GeV/c ^{2}) of the outgoing proton remnant(s)
- double ptmax

  Maximal transverse momentum of the single outgoing leptons.
- double ptmin

  Minimal transverse momentum of the single outgoing leptons.
- double q2max

  The maximal value of Q^2.
- double q2min

  The minimal value of Q^2.
- 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.

- double 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.
- double wmin

  The minimal s on which the cross section is integrated.

5.3.1 Field Documentation

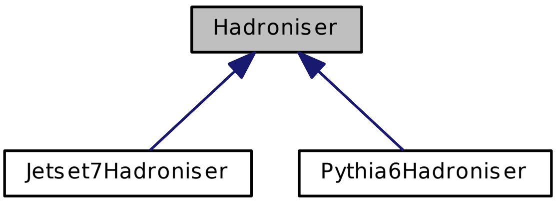
5.3.1.1 int GamGamKinematics::kinematics

Type of kinematics to consider for the process. Can either be :

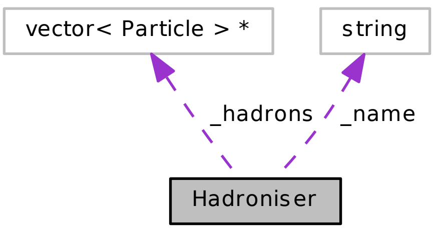
- 0 for the electron-electron elastic case
- 1 for the proton-proton elastic case
- 2 for the proton-proton single-dissociative (or inelastic) case
- 3 for the proton-proton double-dissociative case

5.4 Hadroniser Class Reference

Inheritance diagram for Hadroniser:



Collaboration diagram for Hadroniser:



Public Member Functions

- std::vector< Particle > GetHadrons ()
- bool Hadronise (Particle *part_)

  Main caller to hadronise a particle.
- bool Hadronise (Event *ev_)

  Hadronises a full event.

Protected Attributes

- std::vector< Particle > * _hadrons

  List of hadrons produced by this hadronisation process.
- std::string _name

  Name of the hadroniser.

5.4.1 Detailed Description

Class template to define any hadroniser as a general object with defined methods

Author

Laurent Forthomme `laurent.forthomme@uclouvain.be`

Date

January 2014

5.4.2 Member Function Documentation

5.4.2.1 std::vector<**Particle**> Hadroniser::GetHadrons ( ) [inline]

Gets the full list of hadrons (as Particle objects) produced by the hadronisation

Returns

A vector of Particle containing all the hadrons produced

5.4.2.2 bool Hadroniser::Hadronise (**Event** * ev_) [inline]

Launches the hadroniser on the full event information

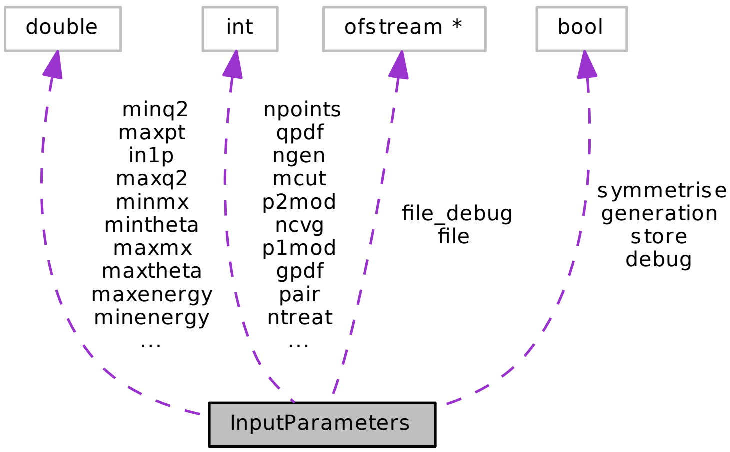
Parameters

*ev_*	The event to hadronise

5.5 InputParameters Class Reference

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

Collaboration diagram for InputParameters:



Public Member Functions

- void Dump ()

  Dumps the input parameters in the console.
- bool ReadConfigFile (std::string inFile_)

  Reads content from config file to load the variables.
- void SetEtaRange (double etamin_, double etamax_)

  Sets the pseudo-rapidity range for the produced leptons.
- bool StoreConfigFile (std::string outFile_)

  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.
- std::ofstream ∗ **file_debug**
- bool generation

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

  PDFLIB group to use.
- double in1p

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

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

  Maximal number of iterations to perform by VEGAS.
- double maxenergy

Maximal energy of the outgoing leptons.
- int maxgen

  Maximal number of events to generate in this run.
- double maxmx

  Maximal M_X of the outgoing proton remnants.
- double maxpt

  Maximal p_T of the outgoing leptons.
- double maxq2

  Maximal value of Q^2, the internal photons lines' virtuality.
- double maxtheta

  Maximal polar angle θ of the outgoing leptons.
- int mcut

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

  Minimal energy of the outgoing leptons.
- double minmx

  Minimal M_X of the outgoing proton remnants.
- double minpt

  Minimal p_T of the outgoing leptons.
- double minq2

  Minimal value of Q^2, the internal photons lines' virtuality.
- double mintheta

  Minimal polar angle θ of the outgoing leptons.
- int **ncvg**
- int ngen

  Number of events already generated in this run.
- int npoints

  Number of points to "shoot" in each integration bin by the algorithm.
- int ntreat

  Maximal number of TREAT calls.
- int p1mod

  First particle's mode.
- int p2mod

  Second particle's mode.
- int pair

  PDG id of the outgoing leptons.
- int qpdf

  Number of quarks.
- int spdf

  PDFLIB set to use.
- bool store

  Are the events generated in this run to be stored in the output file ?
- bool symmetrise

  Control plots objects.

5.5.1 Detailed Description

Note

The default parameters are derived from GMUINI in LPAIR

5.5.2 Member Function Documentation

5.5.2.1 bool InputParameters::ReadConfigFile (std::string inFile_)

Reads the list of parameters to be used in this cross-section computation/events generation from an external input card.

Parameters

*inFile_*	Name of the configuration file to load

Returns

A boolean stating whether this input configuration file is correct or not

5.5.2.2 void InputParameters::SetEtaRange (double etamin_, double etamax_)

Defines the range to cover in pseudo-rapidity for the outgoing leptons produced in this process. This method converts this range into a range in θ, the polar angle.

Parameters

*etamin_*	The minimal value of η for the outgoing leptons
*etamax_*	The maximal value of η for the outgoing leptons

5.5.2.3 bool InputParameters::StoreConfigFile (std::string outFile_)

Parameters

*outFile_*	Name of the configuration file to create

Returns

A boolean stating whether this output configuration file is correctly written or not

5.5.3 Field Documentation

5.5.3.1 bool InputParameters::debug

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

5.5.3.2 double InputParameters::maxmx

Maximal mass of the outgoing proton remnants, M_X, in GeV/c ^{2}.

5.5.3.3 double InputParameters::maxpt

Maximal transverse momentum cut to apply on the outgoing lepton(s)

5.5.3.4 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/c, or
  - 0.75 $< \frac{|p_z|}{|\mathbf{p}|} \leq$ 0.95 and $p_z >$ 1 GeV/c,
- 2 - Cuts on both the outgoing leptons, according to the provided cuts parameters
- 3 - Cuts on at least one outgoing lepton, according to the provided cut parameters

5.5.3.5 double InputParameters::minmx

Minimal mass of the outgoing proton remnants, M_X, in GeV/c ^{2}.

5.5.3.6 double InputParameters::minpt

Minimal transverse momentum cut to apply on the outgoing lepton(s)

5.5.3.7 int InputParameters::ntreat

Note

Is it correctly implemented ?

5.5.3.8 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

5.5.3.9 int InputParameters::p2mod

Note

Was named EMOD in ILPAIR

5.5.3.10 int InputParameters::pair

The particle code of produced leptons, as defined by the PDG convention :

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

5.5.3.11 bool InputParameters::symmetrise

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

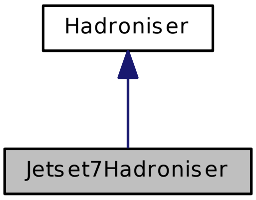
Note

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!

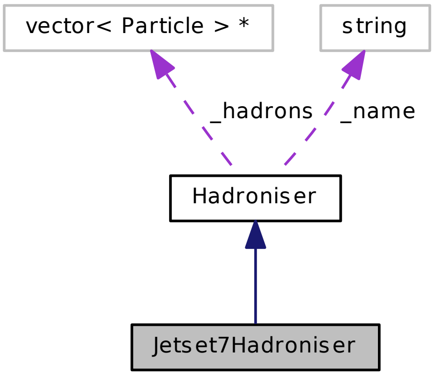
5.6 Jetset7Hadroniser Class Reference

Jetset7 hadronisation algorithm.

Inheritance diagram for Jetset7Hadroniser:



Collaboration diagram for Jetset7Hadroniser:



Public Member Functions

- std::vector< Particle > GetHadrons ()
- bool **Hadronise** (Particle ∗part_)
- bool **Hadronise** (Event ∗ev_)

Protected Attributes

- std::vector< Particle > ∗ _hadrons

  List of hadrons produced by this hadronisation process.
- std::string _name

  Name of the hadroniser.

5.6.1 Member Function Documentation

5.6.1.1 std::vector<**Particle**> Hadroniser::GetHadrons ( ) [inline], [inherited]

Gets the full list of hadrons (as Particle objects) produced by the hadronisation

Returns

A vector of Particle containing all the hadrons produced

5.7 MCGen Class Reference

Core of the Monte-Carlo generator.

Public Member Functions

- MCGen (InputParameters ip_)
  Class constructor.
- 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 **LaunchGeneration** ()
- void **Test** ()

5.7.1 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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 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

5.7.3 Member Function Documentation

5.7.3.1 void MCGen::ComputeXsection (double ∗ , double ∗)

Computes the cross-section for the run defined by this object. This returns the cross-section as well as the absolute error computed along.

5.7.3.2 **InputParameters** MCGen::GetInputParameters ( ) [inline]

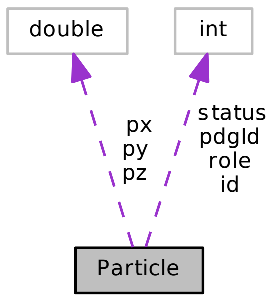
Returns

The InputParameter object embedded in this class

5.8 Particle Class Reference

Kinematics of one particle.

Collaboration diagram for Particle:



Public Member Functions

- Particle (int role_, int pdgId_=0)

  Object constructor (providing the role of the particle in the process, and its Particle Data Group identifier)
- bool AddDaughter (Particle ∗part_)

  Specify a decay product for this particle.
- void Dump ()

  Dumps all the information on this particle.
- void E (double E_)

  Sets the particle's energy.
- double E ()

  Gets the particle's energy.
- double Eta ()

  Pseudo-rapidity.
- std::vector< Particle ∗ > GetDaughters ()

  Gets a vector containing all the daughters from this particle.

- std::string GetLHEline (bool revert_=false)
- Particle ∗ GetMother ()

  Gets the mother particle from which this particle arises.
- bool Hadronise (std::string algo_)

  Hadronises the particle using Pythia.
- double M ()

  Gets the particle's mass.
- bool M (double m_)

  Set the particle's mass in GeV/c ^{2}.
- double M2 ()

  Gets the particle's squared mass.
- unsigned int NumDaughters ()

  Gets the number of daughter particles arising from this one.
- Particle & operator= (const Particle &)

  Copies all the relevant quantities from one Particle object to another.
- bool P (double px_, double py_, double pz_)

  Sets the 3-momentum associated to the particle.
- bool P (double px_, double py_, double pz_, double E_)

  Sets the 4-momentum associated to the particle.
- bool P (double p_[3], double E_)

  Sets the 4-momentum associated to the particle.
- bool P (double p_[4])

  Sets the 4-momentum associated to the particle.
- double P ()

  Norm of the 3-momentum, in GeV/c.
- double ∗ P3 ()

  Returns the particle's 3-momentum.
- double ∗ P4 ()

  Returns the particle's 4-momentum.
- void **PDF2PDG** ()
- double Pt ()

  Transverse momentum, in GeV/c.
- void SetMother (Particle ∗part_)

  Sets the mother particle (from which this particle arises)
- bool Valid ()

  Is this particle a valid particle which can be used for kinematic computations ?

Data Fields

- int id

  Unique identifier of the particle (in a Event object context)
- int pdgId

  Particle Data Group integer identifier.
- 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.
- int status

  Particle status.

5.8.1 Detailed Description

Kinematic information for one particle

5.8.2 Member Function Documentation

5.8.2.1 bool Particle::AddDaughter (**Particle** * part_)

Adds a "daughter" to this particle (one of its decay product(s))

Parameters

*part_*	The Particle object in which this particle will desintegrate or convert

Returns

A boolean stating if the particle has been added to the daughters list or if it was already present before

5.8.2.2 void Particle::Dump ( )

Dumps into the standard output stream all the available information on this particle

5.8.2.3 void Particle::E (double E_) `[inline]`

Parameters

*E_*	Energy, in GeV

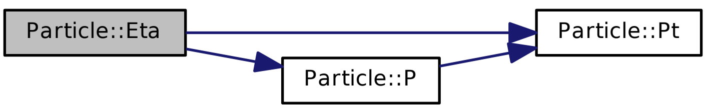
5.8.2.4 double Particle::Eta ( ) `[inline]`

Computes and returns η, the pseudo-rapidity of the particle

Returns

The pseudo-rapidity of the particle

Here is the call graph for this function:



5.8.2.5 std::vector<**Particle***> Particle::GetDaughters ( )

Returns

A Particle objects vector containing all the daughters' kinematic information

5.8.2.6 std::string Particle::GetLHEline (bool revert_ = `false`)

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

Parameters

*revert_*	Is the event symmetric ? If set to true, the third component of the momentum is reverted.

Returns

The LHE line associated to the particle, and containing the particle's history (mother/daughters), its kinematics, and its status

5.8.2.7 bool Particle::Hadronise (std::string algo_)

Hadronises the particle with Pythia, and builds the shower (list of Particle objects) embedded in this object

Parameters

*algo_*	Algorithm in use to hadronise the particle

5.8.2.8 double Particle::M ( ) [inline]

Gets the particle's mass in GeV/c ^{2}.

Returns

The particle's mass

5.8.2.9 bool Particle::P (double px_, double py_, double pz_) [inline]

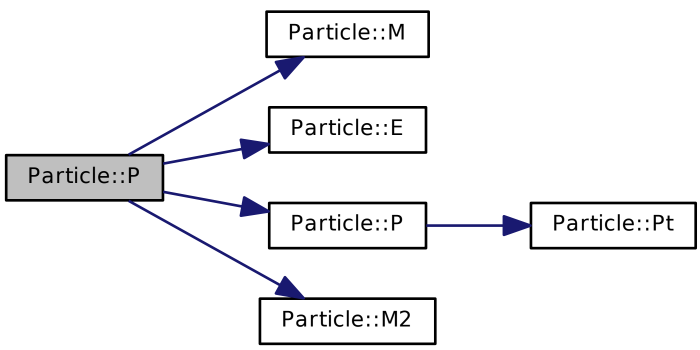
Parameters

*px_*	Momentum along the x-axis, in GeV/c
*py_*	Momentum along the y-axis, in GeV/c
*pz_*	Momentum along the z-axis, in GeV/c

Returns

A boolean stating the validity of this particle (according to its 4-momentum norm)

Here is the call graph for this function:



5.8.2.10 bool Particle::P (double px_, double py_, double pz_, double E_) [inline]

Sets the 4-momentum associated to the particle, and computes its (invariant) mass.

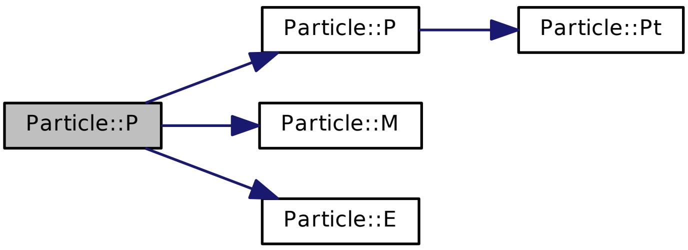
Parameters

*px_*	Momentum along the x-axis, in GeV/c
*py_*	Momentum along the y-axis, in GeV/c
*pz_*	Momentum along the z-axis, in GeV/c
*E_*	Energy, in GeV

Returns

A boolean stating the validity of the particle's kinematics

Here is the call graph for this function:



5.8.2.11 bool Particle::P (double p_[3], double E_)

Parameters

*p_*	3-momentum
*E_*	Energy, in GeV

Returns

A boolean stating the validity of the particle's kinematics

5.8.2.12 bool Particle::P (double p_[4]) [inline]

Parameters

*p_*	4-momentum

Returns

A boolean stating the validity of the particle's kinematics

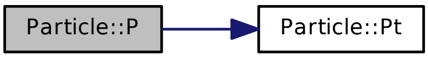
Here is the call graph for this function:

5.8.2.13 double Particle::P ( ) [inline]

Returns

The particle's 3-momentum norm as a double precision float

Here is the call graph for this function:



5.8.2.14 double* Particle::P3 ( ) [inline]

Returns

The particle's 3-momentum as a 3 components double array

5.8.2.15 double* Particle::P4 ( ) [inline]

Builds and returns the particle's 4-momentum as an array ordered as $(\mathbf{p}, E) = (p_x, p_y, p_z, E)$

Returns

The particle's 4-momentum as a 4 components double array

Here is the call graph for this function:

5.8.2.16 void Particle::SetMother (**Particle** * part_)

Sets the "mother" of this particle (particle from which this particle is issued)

Parameters

*part_*	A Particle object containing all the information on the mother particle

5.8.3 Field Documentation

5.8.3.1 int Particle::pdgId

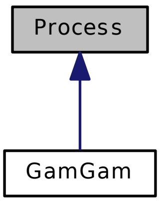
Unique identifier for a particle type. From [1] : *The Monte Carlo particle numbering scheme [...] is intended to facilitate interfacing between event generators, detector simulators, and analysis packages used in particle physics.*

5.8.3.2 int Particle::status

Codes 1-10 correspond to currently existing partons/particles, and larger codes contain partons/particles which no longer exist, or other kinds of event information

5.9 Process Class Reference

Inheritance diagram for Process:



Public Member Functions

- double ComputeWeight ()

  Returns the weight for this point in the phase-space.

5.9.1 Detailed Description

Class template to define any process to compute using this MC integrator/generator

Author

Laurent Forthomme `laurent.forthomme@uclouvain.be`

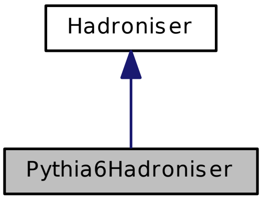
Date

January 2014

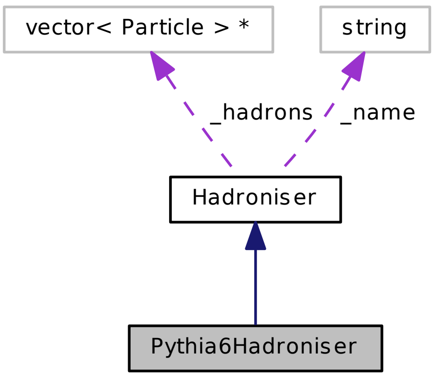
5.10 Pythia6Hadroniser Class Reference

Pythia6 hadronisation algorithm.

Inheritance diagram for Pythia6Hadroniser:



Collaboration diagram for Pythia6Hadroniser:



Public Member Functions

- std::vector< Particle > GetHadrons ()
- bool **Hadronise** (Particle ∗part_)
- bool **Hadronise** (Event ∗ev_)

Protected Attributes

- std::vector< Particle > ∗ _hadrons

  List of hadrons produced by this hadronisation process.
- std::string _name

  Name of the hadroniser.

5.10.1 Member Function Documentation

5.10.1.1 std::vector<**Particle**> Hadroniser::GetHadrons ( ) [inline], [inherited]

Gets the full list of hadrons (as Particle objects) produced by the hadronisation

Returns

A vector of Particle containing all the hadrons produced

5.11 Vegas Class Reference

Vegas Monte-Carlo integrator instance.

Public Member Functions

- Vegas (const int dim_, double f_(double ∗, size_t, void ∗), InputParameters ∗inParam_)
- ∼Vegas ()

    Class destructor.
- void **DumpGrid** ()
- void **Generate** ()
- int Integrate (double ∗result_, double ∗abserr_)

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

    Launches the generation of events.
- int MyIntegrate (double ∗result_, double ∗abserr_)

5.11.1 Constructor & Destructor Documentation

5.11.1.1 Vegas::Vegas (const 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. [2]

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)

5.11.2 Member Function Documentation

5.11.2.1 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

5.11.2.2 int Vegas::LaunchGeneration ( )

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

5.11.2.3 int Vegas::MyIntegrate (double * result_, double * abserr_)

Vegas algorithm to perform the (_dim)-dimensional Monte Carlo integration of a given function as described in [2]

Author

Primary author : G.P. Lepage
This C++ implementation : L. Forthomme

Date

September 1976
Reviewed in Apr 1978
FTN5 version 21 Aug 1984
This C++ implementation is from 12 Dec 2013

Parameters

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

References

[1] J. Beringer et al. Review of Particle Physics (RPP). *Phys.Rev.*, D86:010001, 2012. 27

[2] G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192 – 203, 1978. 1, 29, 30

[3] J.A.M. Vermaseren. Two-photon processes at very high energies. *Nuclear Physics B*, 229(2):347 – 371, 1983. 1, 6

Index