LPAIR++

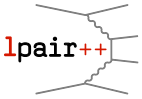
0.2

Generated by Doxygen 1.8.6

Sat Mar 8 2014 00:36:31

Contents

1 Principles



This Monte Carlo generator, based on the LPAIR code developed in the early 1990s by J. Vermaseren *et al*[5], 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[3].

2 Todo List

Global GamGam::ComputeWeight (int nm_=1)

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

Global GamGam::GamGam (int nOpt_=0)

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

3 Deprecated List

Global MCGen::LaunchGeneration ()

This method is to be suppressed since the events generation can now be launched one event at a time using the *GenerateOneEvent* method

4 Hierarchical Index

4.1 Class Hierarchy

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

5 Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

6 File Index

6.1 File List

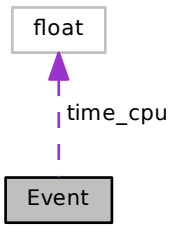
Here is a list of all files with brief descriptions:

7 Data Structure Documentation

7.1 Event Class Reference

Kinematic information on the particles in the event.

Collaboration diagram for Event:



Public Member Functions

- Event ()
- ~Event ()
- int AddParticle (Particle *part_, bool replace_=false)

  Add a particle to the event.
- int AddParticle (int role_, bool replace_=false)
- void clear ()
- void Dump (bool stable_=false)
- Particle * GetById (int id_)

  Gets one particle by its unique identifier in the event.
- Particles GetByIds (std::vector< int > ids_)

  Gets a vector of particles by their unique identifier in the event.
- Particles GetByRole (int role_)

  Gets a list of particles by their role in the event.
- Particles GetDaughters (Particle *part_)

  Gets a vector containing all the daughters from a particle.
- std::string GetLHERecord (const double weight_=1.)

  Gets the LHE block for this event.
- Particle * GetMother (Particle *part_)
- Particle * GetOneByRole (int role_)
- Particles GetParticles ()

  Gets a vector of particles in the event.
- std::vector< int > GetRoles ()
- Particles GetStableParticles ()

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

  Number of particles in the event.
- Event & operator= (const Event &)

  Copies all the relevant quantities from one Event object to another.
- void Store (std::ofstream *, double weight_=1.)

Data Fields

- float time_cpu

7.1.1 Detailed Description

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

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Event::Event ( )

7.1.2.2 Event::~Event ( )

7.1.3 Member Function Documentation

7.1.3.1 int Event::AddParticle (**Particle** * part_, bool replace_ = `false`)

Sets the information on one particle in the process

Parameters

in	*part_*	The Particle object to insert or modify in the event
in	*replace_*	Do we replace the particle if already present in the event or do we append another particle with the same role ?

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

7.1.3.2 int Event::AddParticle (int role_, bool replace_ = `false`)

7.1.3.3 void Event::clear ( ) `[inline]`

7.1.3.4 void Event::Dump (bool stable_ = `false`)

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

Parameters

in	*stable_*	Do we only show the stable particles in this event ?

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

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

Parameters

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

Returns

A pointer to the requested Particle object

7.1.3.6 **Particles** Event::GetByIds (std::vector< int > ids_) `[inline]`

Returns the pointers to the Particle objects corresponding to the unique identifiers in the event

Parameters

in	*ids_*	The unique identifiers to the particles to be selected in the event

Returns

A vector of pointers to the requested Particle objects

7.1.3.7 **Particles** Event::GetByRole (int role_)

Returns the list of pointers to the Particle objects corresponding to a certain role in the process kinematics

Parameters

in	*role_*	The role the particles have to play in the process

Returns

A vector of pointers to the requested Particle objects

7.1.3.8 **Particles** Event::GetDaughters (**Particle** ∗ part_) `[inline]`

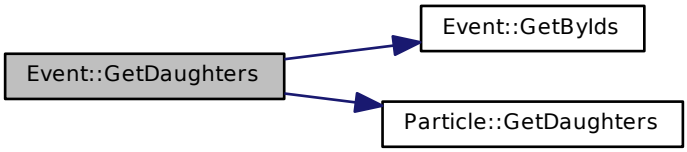
Parameters

in	*part_*	The particle for which the daughter particles have to be retrieved

Returns

A Particle objects vector containing all the daughters' kinematic information

Here is the call graph for this function:



7.1.3.9 std::string Event::GetLHERecord (const double weight_ = 1.)

Returns an event block in a LHE format (a XML-style) with all the information on the particles composing this event

Parameters

in	*weight_*	The weight of the event

Returns

A string containing the kinematic quantities for each of the particles in the event, formatted as the LHE standard requires.

7.1.3.10 **Particle*** Event::GetMother (**Particle** * part_) [inline]

Returns the pointer to the mother particle of any given Particle object in this event

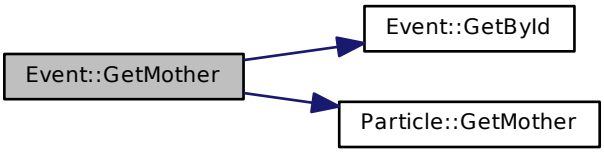
Parameters

in	*part_*	The pointer to the Particle object from which we want to extract the mother particle

Returns

A pointer to the mother Particle object

Here is the call graph for this function:



7.1.3.11 **Particle*** Event::GetOneByRole (int role_) [inline]

Returns the first Particle object in the particles list whose role corresponds to the given argument

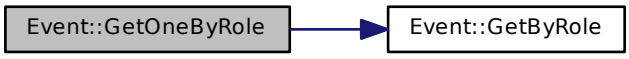
Parameters

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

Returns

A Particle object corresponding to the first particle found in this event

Here is the call graph for this function:



7.1.3.12 **Particles** Event::GetParticles ( )

Returns

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

7.1.3.13 std::vector<int> Event::GetRoles ( )

Gets a list of roles for the given event (really process-dependant for the central system)

Returns

A vector of integers corresponding to all the roles the particles can play in the event

7.1.3.14 **Particles** Event::GetStableParticles ( )

Returns

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

7.1.3.15 int Event::NumParticles ( ) [inline]

Returns

The number of particles in the event, as an integer

7.1.3.16 **Event**& Event::operator= (const **Event** &)

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

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

Parameters

in	*weight_*	The weight of the event

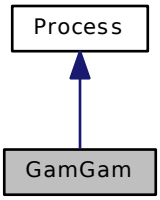
7.1.4 Field Documentation

7.1.4.1 float Event::time_cpu

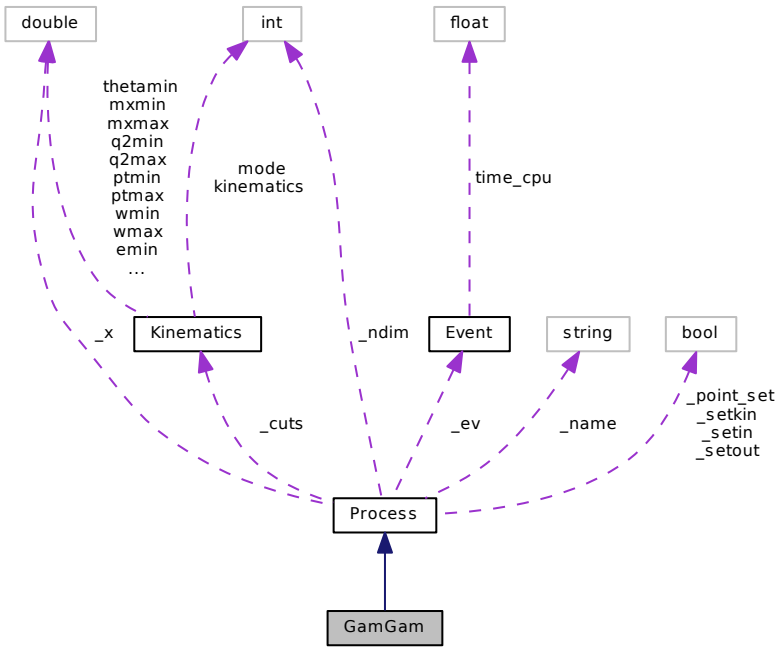
7.2 GamGam Class Reference

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

Inheritance diagram for GamGam:



Collaboration diagram for GamGam:



Public Member Functions

- GamGam (int nOpt_=0)

  Class constructor.
- ~GamGam ()
- void ComputeCMenergy ()

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

  Computes the ougoing proton remnant mass.
- double ComputeWeight (int nm_=1)

  Computes the process' weight for the given point.
- virtual void DumpPoint ()
- void FillKinematics (bool)

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

  Returns the event content (list of particles with an assigned role)
- virtual std::string GetName ()
- 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 ()
- virtual bool IsKinematicsDefined ()

  Is the system's kinematics well defined?
- virtual unsigned int ndim () const
- void PrepareHadronisation (Particle *part_)
- bool SetIncomingParticles (Particle, Particle)

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

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

  Sets the PDG id for the outgoing particles.
- virtual void SetPoint (const unsigned int ndim_, double x_[])

  Sets the phase space point to compute.
- void StoreEvent (std::ofstream *, double)
- virtual double x (const unsigned int idx_)

Protected Attributes

- Event * _ev

  Event object containing all the information on the in- and outgoing particles.
- std::string _name
- bool _point_set

  Is the phase space point set ?
- bool _setin

  Are the event's incoming particles set ?
- bool _setkin

  Is the full event's kinematic set ?
- bool _setout

  Are the event's outgoing particles set ?

7.2.1 Detailed Description

Full class of methods and objects to compute the full analytic matrix element [5] for the $\gamma\gamma \to \ell^+\ell^-$ process according to a set of kinematic constraints provided for the incoming and outgoing particles (the GamGam-Kinematics object). The particle roles in this process are defined as following :

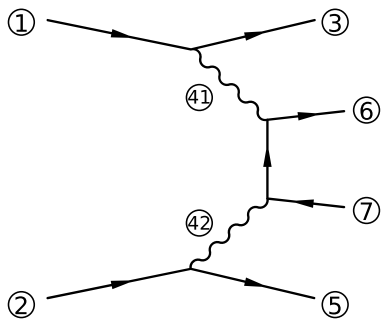


Figure 1: Detailed particle roles in the two-photon

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).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 GamGam::GamGam (int nOpt_ = 0)

Sets the mandatory parameters used in the methods computing the kinematics and the cross-section for the $\gamma\gamma \to \ell^+\ell^-$ process

Parameters

in	*nOpt_*	Optimisation???

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

7.2.2.2 GamGam::~GamGam ( )

7.2.3 Member Function Documentation

7.2.3.1 void GamGam::ComputeCMenergy ( )

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

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

Computes the mass of the outgoing proton remnant if any

Parameters

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

Returns

The mass of the outgoing proton remnant

7.2.3.3 double GamGam::ComputeWeight (int nm_ = 1) [virtual]

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

Parameters

in	*nm_*	???

Returns

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

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

Implements Process.

7.2.3.4 virtual void Process::DumpPoint ( ) [virtual], [inherited]

7.2.3.5 void GamGam::FillKinematics (bool symmetrise_) [virtual]

Fills the private Event object with all the Particle object contained in this event.

Parameters

in	*symmetrise_*	Do we have to symmetrise the event (randomise the production of the positively- and negatively-charged lepton) ?

Reimplemented from Process.

7.2.3.6 double GamGam::GetD3 ( ) [inline]

7.2.3.7 virtual **Event*** Process::GetEvent ( ) [inline], [virtual], [inherited]

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

7.2.3.8 virtual std::string Process::GetName ( ) [inline], [virtual], [inherited]

7.2.3.9 double GamGam::GetS1 ( ) [inline]

7.2.3.10 double GamGam::GetS2 ( ) [inline]

7.2.3.11 double GamGam::GetT1 ( ) [inline]

Returns the value for the first photon virtuality

Returns

t_1, the first photon virtuality

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

Returns the two limit values for the first photon virtuality

Parameters

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

7.2.3.13 double GamGam::GetT2 ( ) [inline]

Returns the value for the second photon virtuality

Returns

t_2, the second photon virtuality

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

Returns the two limit values for the second photon virtuality

Parameters

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

7.2.3.15 double GamGam::GetU1 ( ) [inline]

7.2.3.16 double GamGam::GetU2 ( ) [inline]

7.2.3.17 double GamGam::GetV1 ( ) [inline]

7.2.3.18 double GamGam::GetV2 ( ) [inline]

7.2.3.19 virtual bool Process::IsKinematicsDefined ( ) [inline], [virtual], [inherited]

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

7.2.3.20 virtual unsigned int Process::ndim ( ) const [inline], [virtual], [inherited]

7.2.3.21 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

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

7.2.3.22 bool GamGam::SetIncomingParticles (**Particle** ip1_, **Particle** ip2_) [virtual]

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

Parameters

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

Returns

A boolean stating whether or not the incoming kinematics is properly set for this event

Reimplemented from Process.

7.2.3.23 void GamGam::SetKinematics (**Kinematics** cuts_) [virtual]

Parameters

in	*cuts_*	The Cuts object containing the kinematic parameters

Reimplemented from Process.

7.2.3.24 bool GamGam::SetOutgoingParticles (int part_, int pdgId_) [virtual]

Parameters

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

Returns

A boolean stating whether or not the outgoing kinematics is properly set for this event

Reimplemented from Process.

7.2.3.25 virtual void Process::SetPoint (const unsigned int ndim_, double x_[]) [virtual], [inherited]

Sets the phase space point to compute the weight associated to it.

Parameters

in	*ndim_*	The number of dimensions of the point in the phase space
in	*x_[]*	The (*ndim_*)-dimensional point in the phase space on which the kinematics and the cross-section are computed

7.2.3.26 void GamGam::StoreEvent (std::ofstream * , double)

7.2.3.27 virtual double Process::x (const unsigned int idx_) [inline], [virtual], [inherited]

7.2.4 Field Documentation

7.2.4.1 **Event*** Process::_ev [protected], [inherited]

7.2.4.2 std::string Process::_name [protected], [inherited]

7.2.4.3 bool Process::_point_set [protected], [inherited]

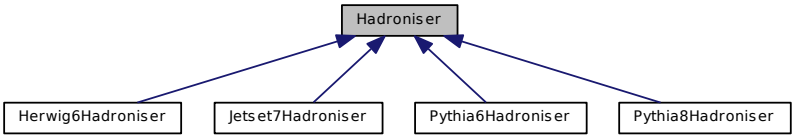
7.2.4.4 bool Process::_setin [protected], [inherited]

7.2.4.5 bool Process::_setkin [protected], [inherited]

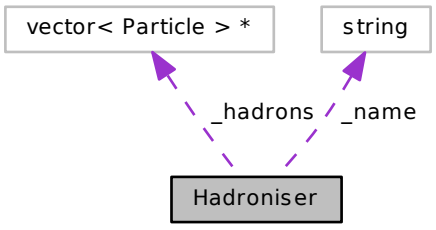
7.2.4.6 bool Process::_setout [protected], [inherited]

7.3 Hadroniser Class Reference

Inheritance diagram for Hadroniser:



Collaboration diagram for Hadroniser:



Public Member Functions

- Hadroniser ()

- ~Hadroniser ()
- std::vector< Particle > GetHadrons ()
- std::string GetName ()
- virtual bool Hadronise (Particle *part_)

  Main caller to hadronise a particle.
- virtual 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.

7.3.1 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

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Hadroniser::Hadroniser ( )

7.3.2.2 Hadroniser::~Hadroniser ( )

7.3.3 Member Function Documentation

7.3.3.1 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

7.3.3.2 std::string Hadroniser::GetName ( ) [inline]

7.3.3.3 virtual bool Hadroniser::Hadronise (**Particle** * part_) [inline], [virtual]

Reimplemented in Pythia6Hadroniser, and Jetset7Hadroniser.

7.3.3.4 virtual bool Hadroniser::Hadronise (**Event** * ev_) [inline], [virtual]

Launches the hadroniser on the full event information

Parameters

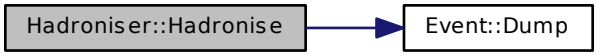
in,out	*ev_*	The event to hadronise

Returns

A boolean stating whether or not the hadronisation occured successfully

Reimplemented in Pythia6Hadroniser, Herwig6Hadroniser, Jetset7Hadroniser, and Pythia8Hadroniser.

Here is the call graph for this function:



7.3.4 Field Documentation

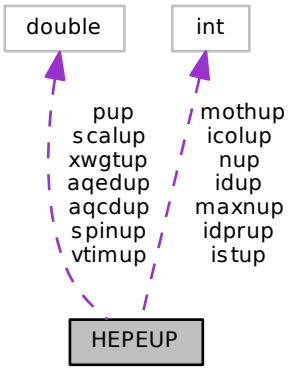
7.3.4.1 std::vector<**Particle**>* Hadroniser::_hadrons [protected]

7.3.4.2 std::string Hadroniser::_name [protected]

7.4 HEPEUP Class Reference

User-process event information.

Collaboration diagram for HEPEUP:



Public Member Functions

- HEPEUP (const int nup_=500)
- ~HEPEUP ()

Data Fields

- double aqcdup

  QCD coupling $\alpha_{\rm QCD}$ used for this event.
- double aqedup

  QED coupling $\alpha_{\rm QED}$ used for this event.
- int * icolup [2]

  Index for the colour flow line passing through the colour (resp. anti-colour) of the particle.
- int idprup

  ID of the process in this event.
- int * idup

  Particle ID according to the Particle Data Group convention.
- int * istup

  Status code.
- int * mothup [2]

  Index of first and last mother.
- int nup

  Number of particle entries in this event.
- double * pup [5]

  Lab-frame momentum of the particle, in GeV.
- double scalup

  Scale of the event in GeV, as used for the calculation of PDFs.
- double * spinup

  Cosine of the angle between the spin-vector of the particle and the 3-momentum of the decaying particle, in the lab frame.
- double * vtimup

  Invariant lifetime $c\tau$ in mm.
- double xwgtup

  Event weight.

Static Public Attributes

- static const int maxnup = 500

  Maximum number of particle entries.

7.4.1 Constructor & Destructor Documentation

7.4.1.1 HEPEUP::HEPEUP (const int nup_ = 500)

7.4.1.2 HEPEUP::~HEPEUP ( )

7.4.2 Field Documentation

7.4.2.1 double HEPEUP::aqcdup

7.4.2.2 double HEPEUP::aqedup

7.4.2.3 int* HEPEUP::icolup[2]

7.4.2.4 int HEPEUP::idprup

7.4.2.5 int* HEPEUP::idup

7.4.2.6 int* HEPEUP::istup

7.4.2.7 const int HEPEUP::maxnup = 500 [static]

7.4.2.8 int* HEPEUP::mothup[2]

7.4.2.9 int HEPEUP::nup

7.4.2.10 double* HEPEUP::pup[5]

7.4.2.11 double HEPEUP::scalup

7.4.2.12 double* HEPEUP::spinup

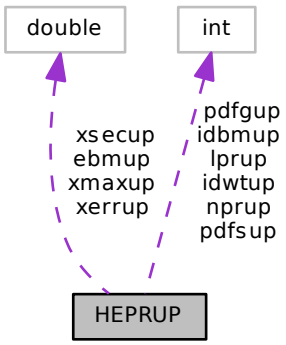
7.4.2.13 double* HEPEUP::vtimup

7.4.2.14 double HEPEUP::xwgtup

7.5 HEPRUP Class Reference

Generic user-process interface for events generator.

Collaboration diagram for HEPRUP:



Public Member Functions

- HEPRUP (const int nprup_=1)
- ~HEPRUP ()

Data Fields

- double ebmup [2]

  Energy in GeV of the beam 1 and 2 particles.
- int idbmup [2]

  ID of the beam 1 and 2 particles according to the Particle Data Group convention.
- int idwtup
- int * lprup
- int nprup
- int pdfgup [2]

  Author group for beam 1 and 2, according to PDFLIB.

- int pdfsup [2]

  PDF set ID for beam 1 and 2, according to PDFLIB.
- double * xerrup
- double * xmaxup
- double * xsecup

7.5.1 Detailed Description

User-process run information

7.5.2 Constructor & Destructor Documentation

7.5.2.1 HEPRUP::HEPRUP (const int nprup_ = 1)

7.5.2.2 HEPRUP::~HEPRUP ( )

7.5.3 Field Documentation

7.5.3.1 double HEPRUP::ebmup[2]

7.5.3.2 int HEPRUP::idbmup[2]

7.5.3.3 int HEPRUP::idwtup

7.5.3.4 int* HEPRUP::lprup

7.5.3.5 int HEPRUP::nprup

7.5.3.6 int HEPRUP::pdfgup[2]

7.5.3.7 int HEPRUP::pdfsup[2]

7.5.3.8 double* HEPRUP::xerrup

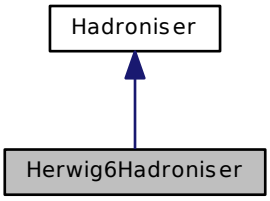
7.5.3.9 double* HEPRUP::xmaxup

7.5.3.10 double* HEPRUP::xsecup

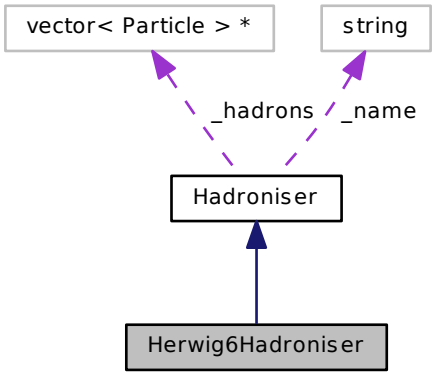
7.6 Herwig6Hadroniser Class Reference

Herwig6 hadronisation algorithm.

Inheritance diagram for Herwig6Hadroniser:



Collaboration diagram for Herwig6Hadroniser:



Public Member Functions

- Herwig6Hadroniser ()
- ~Herwig6Hadroniser ()
- std::vector< Particle > GetHadrons ()
- std::string GetName ()
- virtual 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.

7.6.1 Constructor & Destructor Documentation

7.6.1.1 Herwig6Hadroniser::Herwig6Hadroniser ( )

7.6.1.2 Herwig6Hadroniser::~Herwig6Hadroniser ( )

7.6.2 Member Function Documentation

7.6.2.1 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

7.6.2.2 std::string Hadroniser::GetName ( ) [inline], [inherited]

7.6.2.3 virtual bool Hadroniser::Hadronise (**Particle** * part_) [inline], [virtual], [inherited]

Reimplemented in Pythia6Hadroniser, and Jetset7Hadroniser.

7.6.2.4 bool Herwig6Hadroniser::Hadronise (**Event** * ev_) [virtual]

Launches the hadroniser on the full event information

Parameters

in,out	*ev_*	The event to hadronise

Returns

A boolean stating whether or not the hadronisation occured successfully

Reimplemented from Hadroniser.

7.6.3 Field Documentation

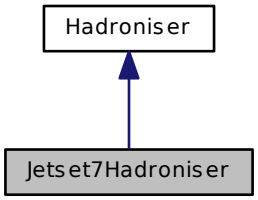
7.6.3.1 std::vector<**Particle**>* Hadroniser::_hadrons [protected], [inherited]

7.6.3.2 std::string Hadroniser::_name [protected], [inherited]

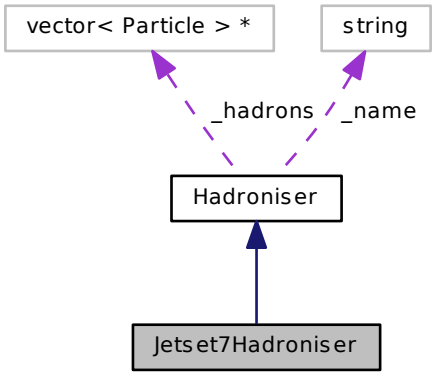
7.7 Jetset7Hadroniser Class Reference

Jetset7 hadronisation algorithm.

Inheritance diagram for Jetset7Hadroniser:



Collaboration diagram for Jetset7Hadroniser:



Public Member Functions

- Jetset7Hadroniser ()
- ~Jetset7Hadroniser ()
- std::vector< Particle > GetHadrons ()
- std::string GetName ()
- 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.

7.7.1 Constructor & Destructor Documentation

7.7.1.1 Jetset7Hadroniser::Jetset7Hadroniser ( )

7.7.1.2 Jetset7Hadroniser::~Jetset7Hadroniser ( )

7.7.2 Member Function Documentation

7.7.2.1 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

7.7.2.2 std::string Hadroniser::GetName ( ) [inline], [inherited]

7.7.2.3 bool Jetset7Hadroniser::Hadronise (**Particle** * part_) [virtual]

Reimplemented from Hadroniser.

7.7.2.4 bool Jetset7Hadroniser::Hadronise (**Event** * ev_) [virtual]

Launches the hadroniser on the full event information

Parameters

in,out	ev_	The event to hadronise

Returns

A boolean stating whether or not the hadronisation occured successfully

Reimplemented from Hadroniser.

7.7.3 Field Documentation

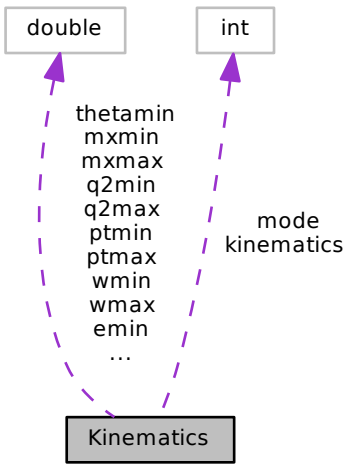
7.7.3.1 std::vector<**Particle**>* Hadroniser::_hadrons [protected], [inherited]

7.7.3.2 std::string Hadroniser::_name [protected], [inherited]

7.8 Kinematics Class Reference

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

Collaboration diagram for Kinematics:



Public Member Functions

- Kinematics ()
- ~Kinematics ()

- void Dump ()

  Dumps all the parameters used in this process cross-section computation / events generation.

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.

7.8.1 Constructor & Destructor Documentation

7.8.1.1 Kinematics::Kinematics ( )

7.8.1.2 Kinematics::∼Kinematics ( )

7.8.2 Member Function Documentation

7.8.2.1 void Kinematics::Dump ( )

7.8.3 Field Documentation

7.8.3.1 double Kinematics::emax

7.8.3.2 double Kinematics::emin

7.8.3.3 int Kinematics::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

7.8.3.4 int Kinematics::mode

7.8.3.5 double Kinematics::mxmax

7.8.3.6 double Kinematics::mxmin

7.8.3.7 double Kinematics::ptmax

7.8.3.8 double Kinematics::ptmin

7.8.3.9 double Kinematics::q2max

7.8.3.10 double Kinematics::q2min

7.8.3.11 double Kinematics::thetamax

7.8.3.12 double Kinematics::thetamin

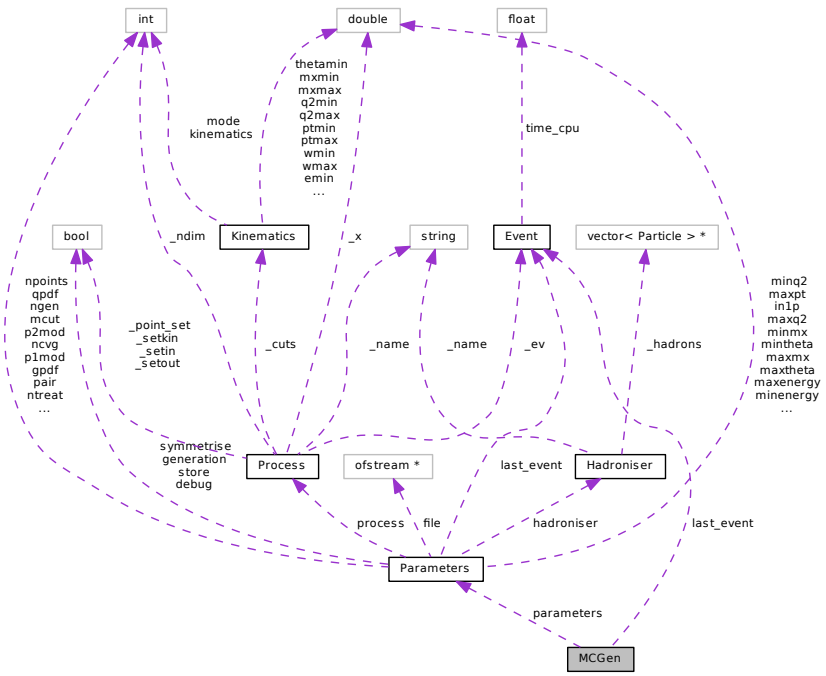
7.8.3.13 double Kinematics::wmax

7.8.3.14 double Kinematics::wmin

7.9 MCGen Class Reference

Core of the Monte-Carlo generator.

Collaboration diagram for MCGen:



Public Member Functions

- MCGen ()

  Class constructor.
- MCGen (Parameters *ip_)

  Class constructor.
- ~MCGen ()
- void AnalyzePhaseSpace (const std::string)

  Returns the set of parameters used to setup the phase space to integrate.
- void ComputeXsection (double *xsec_, double *err_)

  Compute the cross-section for the given process.
- Event * GenerateOneEvent ()
- void LaunchGeneration ()
- void PrintHeader ()
- void Test ()

Data Fields

- Event * last_event

  Last event generated in this run.
- Parameters * parameters

7.9.1 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

7.9.2 Constructor & Destructor Documentation

7.9.2.1 MCGen::MCGen ( )

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

7.9.2.2 MCGen::MCGen (**Parameters** * 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

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

7.9.2.3 MCGen::~MCGen ( )

7.9.3 Member Function Documentation

7.9.3.1 void MCGen::AnalyzePhaseSpace (const std::string)

Returns

The Parameter object embedded in this class

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

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

Parameters

out	*xsec_*	The computed cross-section, in pb
out	*err_*	The absolute integration error on the computed cross-section, in pb

7.9.3.3 **Event*** MCGen::GenerateOneEvent ( )

Generates one single event given the phase space computed by Vegas in the integration step

Returns

A pointer to the Event object generated in this run

7.9.3.4 void MCGen::LaunchGeneration ( )

Launches the full events generation

Deprecated This method is to be suppressed since the events generation can now be launched one event at a time using the *GenerateOneEvent* method

7.9.3.5 void MCGen::PrintHeader ( )

7.9.3.6 void MCGen::Test ( )

7.9.4 Field Documentation

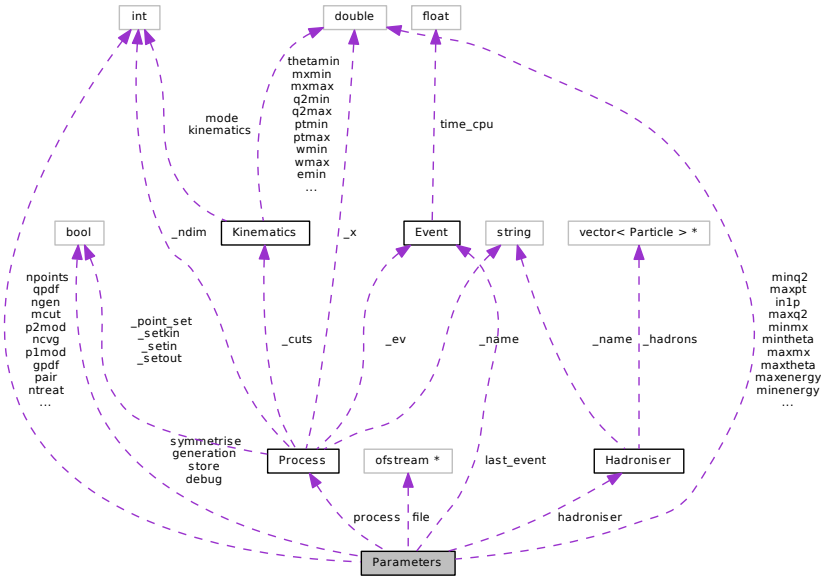
7.9.4.1 **Event*** MCGen::last_event

7.9.4.2 **Parameters*** MCGen::parameters

7.10 Parameters Class Reference

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

Collaboration diagram for Parameters:



Public Member Functions

- Parameters ()
- ~Parameters ()
- 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.
- bool generation

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

  PDFLIB group to use.
- Hadroniser * hadroniser

  Hadronisation algorithm to use for the proton(s) remnants fragmentation.
- 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.
- Event * last_event

  The pointer to the last event produced in this run.
- 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.
- Process * process
- 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.

7.10.1 Detailed Description

Note

The default parameters are derived from GMUINI in LPAIR

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Parameters::Parameters ( )

7.10.2.2 Parameters::∼Parameters ( )

7.10.3 Member Function Documentation

7.10.3.1 void Parameters::Dump ( )

7.10.3.2 bool Parameters::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

in	*inFile_*	Name of the configuration file to load

Returns

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

7.10.3.3 void Parameters::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

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

7.10.3.4 bool Parameters::StoreConfigFile (std::string outFile_)

Parameters

in	*outFile_*	Name of the configuration file to create

Returns

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

7.10.4 Field Documentation

7.10.4.1 bool Parameters::debug

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

7.10.4.2 std::ofstream∗ Parameters::file

7.10.4.3 bool Parameters::generation

7.10.4.4 int Parameters::gpdf

7.10.4.5 **Hadroniser**∗ Parameters::hadroniser

7.10.4.6 double Parameters::in1p

7.10.4.7 double Parameters::in2p

7.10.4.8 int Parameters::itvg

7.10.4.9 **Event**∗ Parameters::last_event

7.10.4.10 double Parameters::maxenergy

7.10.4.11 int Parameters::maxgen

7.10.4.12 double Parameters::maxmx

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

7.10.4.13 double Parameters::maxpt

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

7.10.4.14 double Parameters::maxq2

7.10.4.15 double Parameters::maxtheta

7.10.4.16 int Parameters::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

7.10.4.17 double Parameters::minenergy

7.10.4.18 double Parameters::minmx

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

7.10.4.19 double Parameters::minpt

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

7.10.4.20 double Parameters::minq2

7.10.4.21 double Parameters::mintheta

7.10.4.22 int Parameters::ncvg

7.10.4.23 int Parameters::ngen

7.10.4.24 int Parameters::npoints

7.10.4.25 int Parameters::ntreat

Note

Is it correctly implemented ?

7.10.4.26 int Parameters::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

7.10.4.27 int Parameters::p2mod

Note

Was named EMOD in ILPAIR

7.10.4.28 int Parameters::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

7.10.4.29 **Process**∗ Parameters::process

7.10.4.30 int Parameters::qpdf

7.10.4.31 int Parameters::spdf

7.10.4.32 bool Parameters::store

7.10.4.33 bool Parameters::symmetrise

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

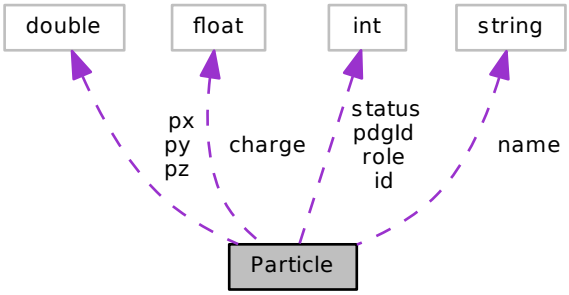
Note

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!

7.11 Particle Class Reference

Kinematics of one particle.

Collaboration diagram for Particle:



Public Member Functions

- Particle ()
- Particle (int role_, int pdgId_=0)

  Object constructor (providing the role of the particle in the process, and its Particle Data Group identifier)
- ~Particle ()
- 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< int > GetDaughters ()

  Gets a vector containing all the daughters unique identifiers from this particle.
- std::string GetLHEline (bool revert_=false)
- int GetMother ()

  Gets the unique identifier to 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.
- bool operator< (const Particle &rhs)

  Comparison operator to enable the sorting of particles in an event according to their unique identifier.
- 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 Phi ()
- bool Primary ()

  Is this particle a primary particle ?
- double Pt ()

  Transverse momentum, in GeV/c.
- double Rapidity ()

  Rapidity.
- 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

- float charge

  The particle's electric charge (given as a float number, for the quarks and bound states)
- int id

  Unique identifier of the particle (in a Event object context)
- std::string name

  Particle's name in a human-readable format.
- 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.

7.11.1 Detailed Description

Kinematic information for one particle

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Particle::Particle ( )

7.11.2.2 Particle::Particle (int role_, int pdgId_ = 0)

7.11.2.3 Particle::~Particle ( )

7.11.3 Member Function Documentation

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

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

Parameters

in	*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

7.11.3.2 void Particle::Dump ( )

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

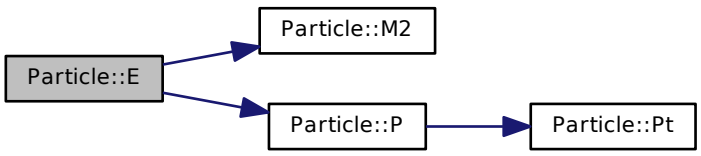
7.11.3.3 void Particle::E (double E_) [inline]

Parameters

in	*E_*	Energy, in GeV

7.11.3.4 double Particle::E ( ) [inline]

Here is the call graph for this function:



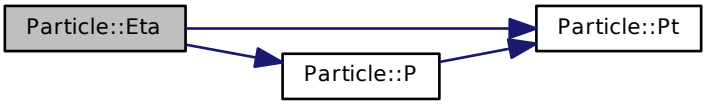
7.11.3.5 double Particle::Eta ( ) [inline]

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:



7.11.3.6 std::vector<int> Particle::GetDaughters ( )

Returns

An integer vector containing all the daughters' unique identifier in the event

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

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

Parameters

`in`	*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

7.11.3.8 int Particle::GetMother ( ) `[inline]`

Returns

An integer representing the unique identifier to the mother of this particle in the event

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

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

Parameters

`in`	*algo_*	Algorithm in use to hadronise the particle

7.11.3.10 double Particle::M ( ) `[inline]`

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

Returns

The particle's mass

7.11.3.11 bool Particle::M (double m_)

7.11.3.12 double Particle::M2 ( ) [inline]

7.11.3.13 unsigned int Particle::NumDaughters ( ) [inline]

7.11.3.14 bool Particle::operator< (const **Particle** & rhs) [inline]

7.11.3.15 **Particle**& Particle::operator= (const **Particle** &)

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

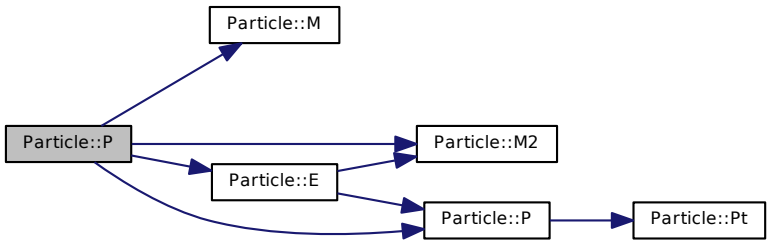
Parameters

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



7.11.3.17 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.

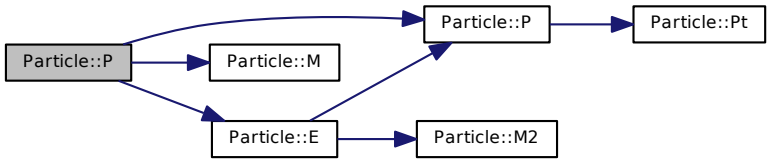
Parameters

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

Returns

A boolean stating the validity of the particle's kinematics

Here is the call graph for this function:



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

Parameters

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

Returns

A boolean stating the validity of the particle's kinematics

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

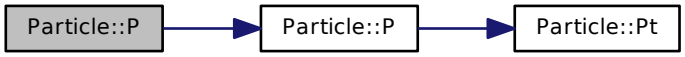
Parameters

in	*p_*	4-momentum

Returns

A boolean stating the validity of the particle's kinematics

Here is the call graph for this function:



7.11.3.20 double Particle::P ( ) [inline]

Returns

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

Here is the call graph for this function:

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

Returns

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

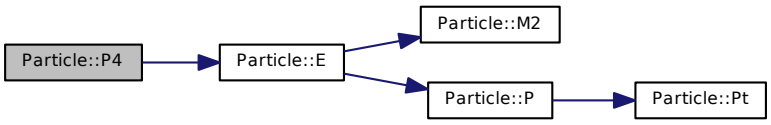
7.11.3.22 double* Particle::P4 ( ) [inline]

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:



7.11.3.23 void Particle::PDF2PDG ( )

7.11.3.24 double Particle::Phi ( ) [inline]

Here is the call graph for this function:

7.11.3.25 bool Particle::Primary ( ) [inline]

7.11.3.26 double Particle::Pt ( ) [inline]

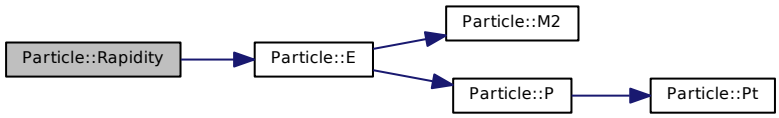
7.11.3.27 double Particle::Rapidity ( ) [inline]

Computes and returns y, the rapidity of the particle

Returns

The rapidity of the particle

Here is the call graph for this function:



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

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

Parameters

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

7.11.3.29 bool Particle::Valid ( )

7.11.4 Field Documentation

7.11.4.1 float Particle::charge

7.11.4.2 int Particle::id

7.11.4.3 std::string Particle::name

7.11.4.4 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.*

7.11.4.5 double Particle::px

7.11.4.6 double Particle::py

7.11.4.7 double Particle::pz

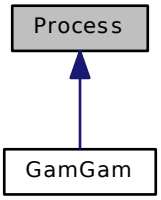
7.11.4.8 int Particle::role

7.11.4.9 int Particle::status

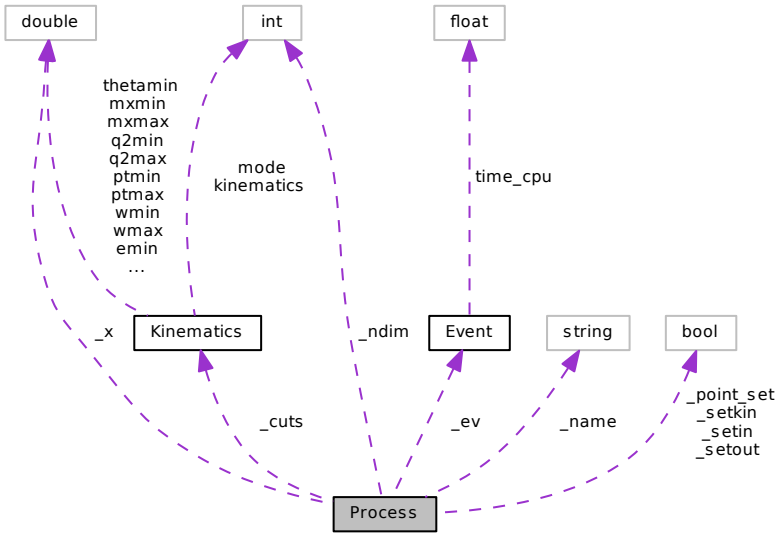
Codes 1-10 correspond to currently existing partons/particles, and larger codes contain partons/particles which no longer exist, or other kinds of event information

7.12 Process Class Reference

Inheritance diagram for Process:



Collaboration diagram for Process:



Public Member Functions

- Process ()
- ~Process ()
- virtual double ComputeWeight (int nm_=1)=0

  Returns the weight for this point in the phase-space.
- virtual void DumpPoint ()
- virtual void FillKinematics (bool symmetrise_=false)

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

  Returns the event content (list of particles with an assigned role)
- virtual std::string GetName ()

- virtual bool IsKinematicsDefined ()

  Is the system's kinematics well defined?
- virtual unsigned int ndim () const
- virtual bool SetIncomingParticles (Particle ip1_, Particle ip2_)

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

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

  Sets the PDG id for the outgoing particles.
- virtual void SetPoint (const unsigned int ndim_, double x_[])

  Sets the phase space point to compute.
- virtual double x (const unsigned int idx_)

Protected Attributes

- Kinematics _cuts

  Set of cuts to apply on the final phase space.
- Event * _ev

  Event object containing all the information on the in- and outgoing particles.
- std::string _name
- unsigned int _ndim
- bool _point_set

  Is the phase space point set ?
- bool _setin

  Are the event's incoming particles set ?
- bool _setkin

  Is the full event's kinematic set ?
- bool _setout

  Are the event's outgoing particles set ?
- double * _x

7.12.1 Detailed Description

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

Author

Laurent Forthomme laurent.forthomme@uclouvain.be

Date

January 2014

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Process::Process ( )

7.12.2.2 Process::~Process ( )

7.12.3 Member Function Documentation

7.12.3.1 virtual double Process::ComputeWeight (int nm_ = 1) [pure virtual]

Implemented in GamGam.

7.12.3.2 virtual void Process::DumpPoint ( ) [virtual]

7.12.3.3 virtual void Process::FillKinematics (bool symmetrise_ = false) [inline], [virtual]

Fills the private Event object with all the Particle object contained in this event.

Parameters

in	*symmetrise_*	Do we have to symmetrise the event (randomise the production of the positively- and negatively-charged lepton) ?

Reimplemented in GamGam.

7.12.3.4 virtual **Event**∗ Process::GetEvent ( ) [inline], [virtual]

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

7.12.3.5 virtual std::string Process::GetName ( ) [inline], [virtual]

7.12.3.6 virtual bool Process::IsKinematicsDefined ( ) [inline], [virtual]

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

7.12.3.7 virtual unsigned int Process::ndim ( ) const [inline], [virtual]

7.12.3.8 virtual bool Process::SetIncomingParticles (**Particle** ip1_, **Particle** ip2_) [inline], [virtual]

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

Parameters

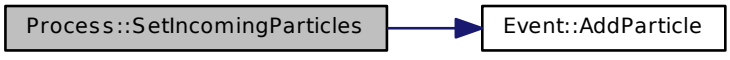
in	*ip1_*	Information on the first incoming particle
in	*ip2_*	Information on the second incoming particle

Returns

A boolean stating whether or not the incoming kinematics is properly set for this event

Reimplemented in GamGam.

Here is the call graph for this function:



7.12.3.9 virtual void Process::SetKinematics (**Kinematics** cuts_) [inline], [virtual]

Parameters

in	*cuts_*	The Cuts object containing the kinematic parameters

Reimplemented in GamGam.

7.12.3.10 virtual bool Process::SetOutgoingParticles (int part_, int pdgId_) [inline], [virtual]

Parameters

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

Returns

A boolean stating whether or not the outgoing kinematics is properly set for this event

Reimplemented in GamGam.

Here is the call graph for this function:

7.12.3.11 virtual void Process::SetPoint (const unsigned int ndim_, double x_[]) [virtual]

Sets the phase space point to compute the weight associated to it.

Parameters

in	*ndim_*	The number of dimensions of the point in the phase space
in	*x_[]*	The (*ndim_*)-dimensional point in the phase space on which the kinematics and the cross-section are computed

7.12.3.12 virtual double Process::x (const unsigned int idx_) [inline], [virtual]

7.12.4 Field Documentation

7.12.4.1 **Kinematics** Process::_cuts [protected]

7.12.4.2 **Event*** Process::_ev [protected]

7.12.4.3 std::string Process::_name [protected]

7.12.4.4 unsigned int Process::_ndim [protected]

7.12.4.5 bool Process::_point_set [protected]

7.12.4.6 bool Process::_setin [protected]

7.12.4.7 bool Process::_setkin [protected]

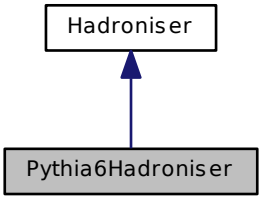
7.12.4.8 bool Process::_setout [protected]

7.12.4.9 double* Process::_x [protected]

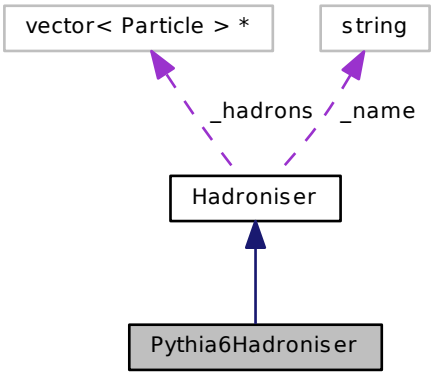
7.13 Pythia6Hadroniser Class Reference

Pythia6 hadronisation algorithm.

Inheritance diagram for Pythia6Hadroniser:



Collaboration diagram for Pythia6Hadroniser:



Public Member Functions

- Pythia6Hadroniser ()
- ~Pythia6Hadroniser ()
- std::vector< Particle > GetHadrons ()
- std::string GetName ()
- 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.

7.13.1 Detailed Description

Full interface to the Pythia6 [4] algorithm. It can be used in a single particle decay mode as well as a full event hadronisation using the string model, as in Jetset.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Pythia6Hadroniser::Pythia6Hadroniser ( )

7.13.2.2 Pythia6Hadroniser::~Pythia6Hadroniser ( )

7.13.3 Member Function Documentation

7.13.3.1 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

7.13.3.2 std::string Hadroniser::GetName ( ) [inline], [inherited]

7.13.3.3 bool Pythia6Hadroniser::Hadronise (**Particle** * part_) [virtual]

Reimplemented from Hadroniser.

7.13.3.4 bool Pythia6Hadroniser::Hadronise (**Event** * ev_) [virtual]

Launches the hadroniser on the full event information

Parameters

in,out	*ev_*	The event to hadronise

Returns

A boolean stating whether or not the hadronisation occured successfully

Reimplemented from Hadroniser.

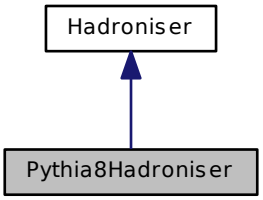
7.13.4 Field Documentation

7.13.4.1 std::vector<**Particle**>* Hadroniser::_hadrons [protected], [inherited]

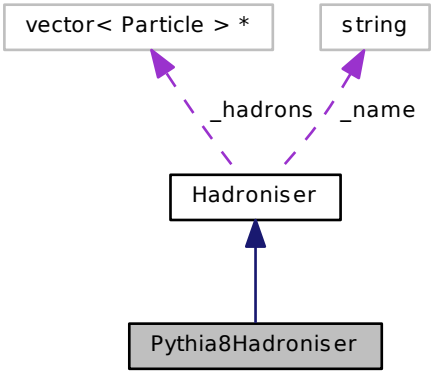
7.13.4.2 std::string Hadroniser::_name [protected], [inherited]

7.14 Pythia8Hadroniser Class Reference

Inheritance diagram for Pythia8Hadroniser:



Collaboration diagram for Pythia8Hadroniser:



Public Member Functions

- Pythia8Hadroniser ()
- ~Pythia8Hadroniser ()
- std::vector< Particle > GetHadrons ()
- std::string GetName ()
- bool Hadronise (Event *ev_)

  Hadronises a full event.
- virtual bool Hadronise (Particle *part_)

  Main caller to hadronise a particle.

Protected Attributes

- std::vector< Particle > * _hadrons

  List of hadrons produced by this hadronisation process.

- std::string _name
  Name of the hadroniser.

7.14.1 Constructor & Destructor Documentation

7.14.1.1 Pythia8Hadroniser::Pythia8Hadroniser ( )

7.14.1.2 Pythia8Hadroniser::~Pythia8Hadroniser ( )

7.14.2 Member Function Documentation

7.14.2.1 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

7.14.2.2 std::string Hadroniser::GetName ( ) [inline], [inherited]

7.14.2.3 bool Pythia8Hadroniser::Hadronise (**Event** * ev_) [virtual]

Launches the hadroniser on the full event information

Parameters

in,out	*ev_*	The event to hadronise

Returns

A boolean stating whether or not the hadronisation occured successfully

Reimplemented from Hadroniser.

7.14.2.4 virtual bool Hadroniser::Hadronise (**Particle** * part_) [inline], [virtual], [inherited]

Reimplemented in Pythia6Hadroniser, and Jetset7Hadroniser.

7.14.3 Field Documentation

7.14.3.1 std::vector<**Particle**>* Hadroniser::_hadrons [protected], [inherited]

7.14.3.2 std::string Hadroniser::_name [protected], [inherited]

7.15 Vegas Class Reference

Vegas Monte-Carlo integrator instance.

Public Member Functions

- Vegas (const int dim_, double f_(double *, size_t, void *), Parameters *inParam_)
- ~Vegas ()
  Class destructor.
- void Generate ()
  Launches the generation of events.
- bool GenerateOneEvent ()
  Generates one single event according to the method defined in the Fortran 77 version of LPAIR.
- int Integrate (double *result_, double *abserr_)

7.15.1 Detailed Description

Main occurence of the Monte-Carlo integrator[3] developed by G.P. Lepage in 1978

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Vegas::Vegas (const int dim_, double f_double *, size_t, void *, **Parameters** * inParam_)

Constructs the class by booking the memory and structures for the Vegas integrator. This code is based on the Vegas Monte Carlo integration algorithm developed by P. Lepage, as documented in [3]

Parameters

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

7.15.2.2 Vegas::~Vegas ( )

7.15.3 Member Function Documentation

7.15.3.1 void Vegas::Generate ( )

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

7.15.3.2 bool Vegas::GenerateOneEvent ( )

Generates one event according to the grid parameters set in Vegas::SetGen

Returns

A boolean stating if the generation was successful (in term of the computed weight for the phase space point)

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

Vegas algorithm to perform the (_dim)-dimensional Monte Carlo integration of a given function as described in [3]

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

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

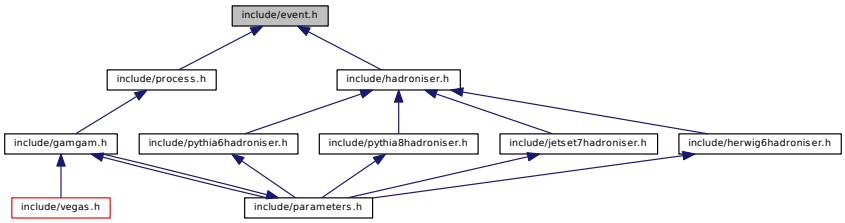
Returns

0 if the integration was performed successfully

8 File Documentation

8.1 include/event.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Event

  Kinematic information on the particles in the event.

Typedefs

- typedef std::vector< Particle * > Particles

  Convention to simplify the user interface while fetching a list of particles in the event.

- typedef std::multimap< int, Particle > ParticlesMap

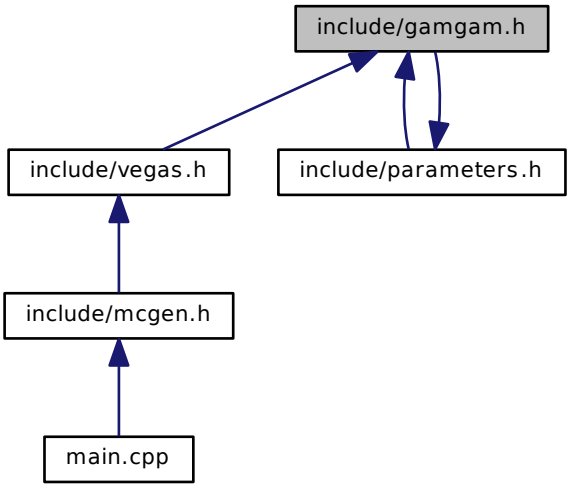
8.1.1 Typedef Documentation

8.1.1.1 typedef std::vector<**Particle***> **Particles**

8.1.1.2 typedef std::multimap<int,**Particle**> **ParticlesMap**

8.2 include/gamgam.h File Reference

This graph shows which files directly or indirectly include this file:



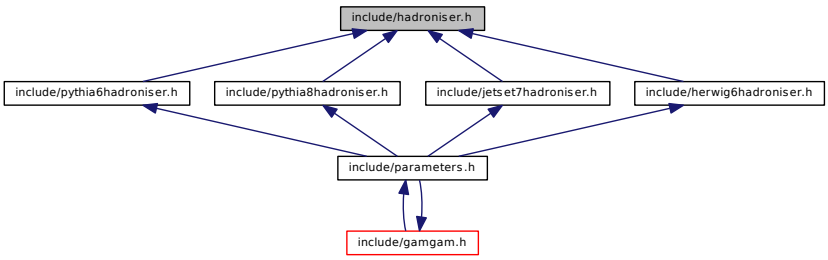
Data Structures

- class GamGam

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

8.3 include/hadroniser.h File Reference

This graph shows which files directly or indirectly include this file:

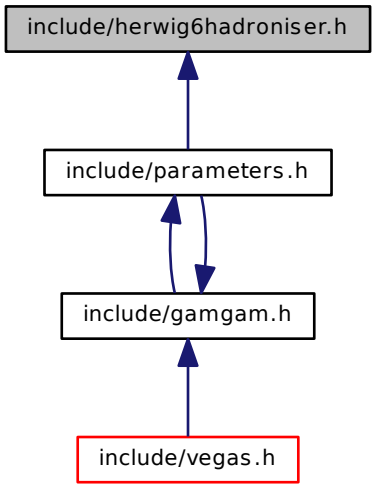


Data Structures

- class Hadroniser

8.4 include/herwig6hadroniser.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Herwig6Hadroniser
  Herwig6 hadronisation algorithm.

Macros

- #define NMXHEP 4000

Functions

- void hwaend_ ()
- void hwdhad_ ()

Variables

- struct {
  int **idhep** [NMXHEP]
  int **isthep** [NMXHEP]
  int **jdahep** [NMXHEP][2]
  int **jmohep** [NMXHEP][2]
  int **nevhep**
  int **nhep**
  double **phep** [NMXHEP][5]
  double **vhep** [NMXHEP][4]
  } hepevt_

8.4.1 Macro Definition Documentation

8.4.1.1 #define NMXHEP 4000

8.4.2 Function Documentation

8.4.2.1 void hwaend_ ( )

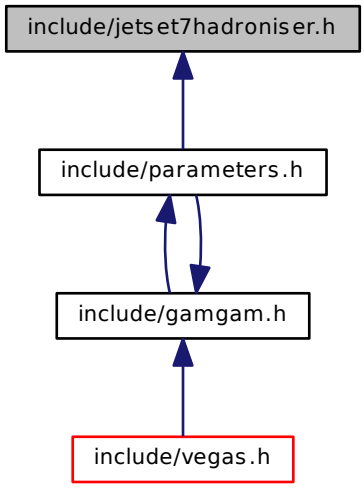
8.4.2.2 void hwdhad_ ( )

8.4.3 Variable Documentation

8.4.3.1 struct { ... } hepevt_

8.5 include/jetset7hadroniser.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Jetset7Hadroniser

  Jetset7 hadronisation algorithm.

Macros

- #define NAME_CHR 16

Functions

- int luchge_ (int &)
- void luexec_ ()
- void lugive_ (const char *, int)
- void lujoin_ (int &, int &)

- void lulist_ (int &)
- void luname_ (int &, char *, int)
- float ulmass_ (int &)

Variables

- struct {
  int **k** [5][4000]
  int **n**
  float **p** [5][4000]
  float **v** [5][4000]
  } lujets_

8.5.1 Macro Definition Documentation

8.5.1.1 #define NAME_CHR 16

8.5.2 Function Documentation

8.5.2.1 int luchge_ (int &)

8.5.2.2 void luexec_ ( )

8.5.2.3 void lugive_ (const char * , int)

8.5.2.4 void lujoin_ (int & , int &)

8.5.2.5 void lulist_ (int &)

8.5.2.6 void luname_ (int & , char * , int)

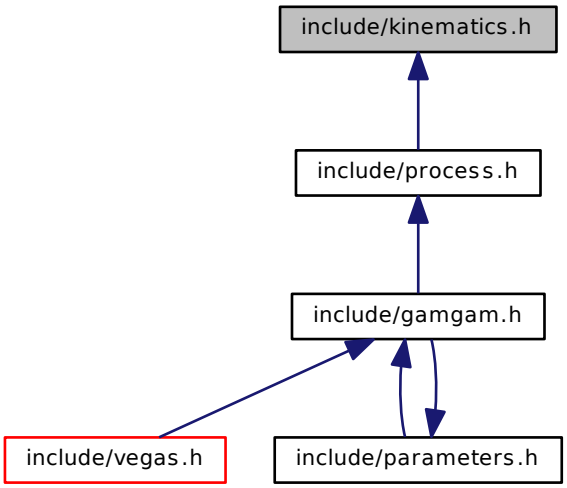
8.5.2.7 float ulmass_ (int &)

8.5.3 Variable Documentation

8.5.3.1 struct { ... } lujets_

8.6 include/kinematics.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Kinematics

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

8.7 include/lheutils.h File Reference

Data Structures

- class HEPEUP

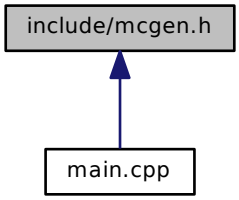
  User-process event information.

- class HEPRUP

  Generic user-process interface for events generator.

8.8 include/mcgen.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class MCGen

  Core of the Monte-Carlo generator.

Functions

- double f (double *, size_t, void *)

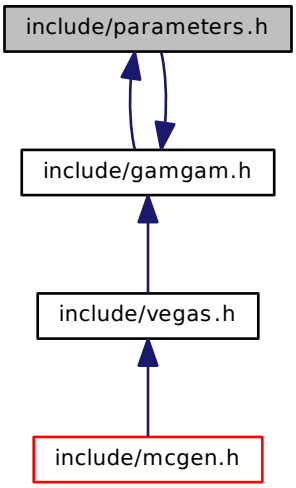
8.8.1 Function Documentation

8.8.1.1 double f (double * , size_t , void *)

The function to be integrated, which returns the value of the weight of an event, including the matrix element of the process, all the kinematic factors, and the cut restrictions. x is an array of random numbers used to select a random point inside the phase space.

8.9 include/parameters.h File Reference

This graph shows which files directly or indirectly include this file:

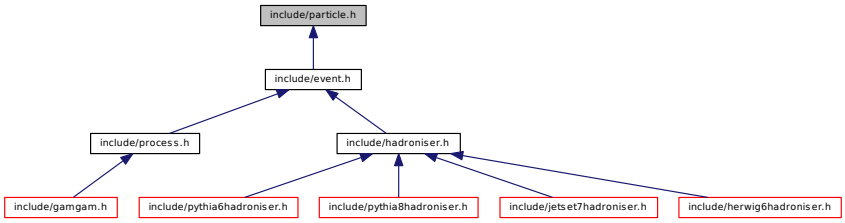


Data Structures

- class Parameters
  List of parameters used to start and run the simulation job.

8.10 include/particle.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Particle
  Kinematics of one particle.

Functions

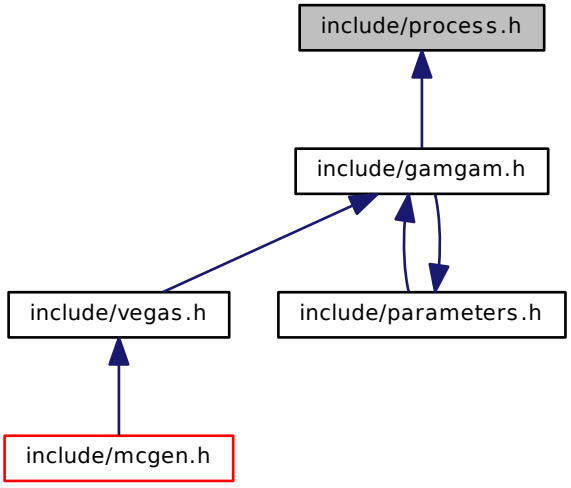
- bool compareParticlePtrs (Particle *a, Particle *b)

8.10.1 Function Documentation

8.10.1.1 bool compareParticlePtrs (**Particle** * a, **Particle** * b) [inline]

8.11 include/process.h File Reference

This graph shows which files directly or indirectly include this file:

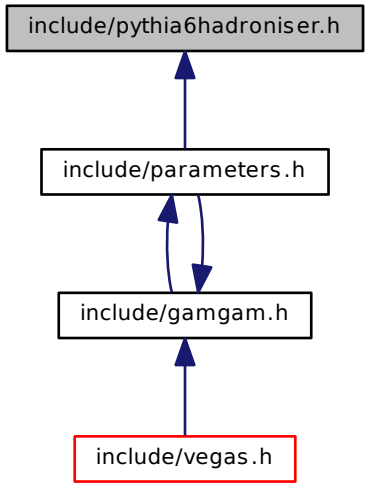


Data Structures

- class Process

8.12 include/pythia6hadroniser.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Pythia6Hadroniser
  Pythia6 hadronisation algorithm.

Macros

- #define NAME_CHR 16
  Maximal number of characters to fetch for the particle's name.

Functions

- void pyckbd_ ()
- void pyexec_ ()
  Launch the Pythia6 fragmentation.
- void pygive_ (const char *, int)
  Set a parameter value to the Pythia6 module.
- void pyjoin_ (int &, int &)
  Joins two coloured particles in a colour singlet.
- void pylist_ (int &)
  Lists all the particles in the event in a human-readable format.
- double pymass_ (int &)
  Get the particle's mass in GeV from the Pythia6 module.
- void pyname_ (int &, char *, int)
  Get a particle's human-readable name from the Pythia6 module.
- double pyp_ (int &, int &)
  Get information on a particle from the Pythia6 module.

Variables

- struct {
  int **k** [5][4000]
  Particles' general information (status, PDG id, mother, daughter 1, daughter 2)
  int **n**
  Number of particles in the event.
  int **npad**
  double **p** [5][4000]
  Particles' kinematics, in GeV (px, py, pz, E, M)
  double **v** [5][4000]
  Primary vertex for the particles.
  } pyjets_

  Particles content of the event.

8.12.1 Macro Definition Documentation

8.12.1.1 #define NAME_CHR 16

8.12.2 Function Documentation

8.12.2.1 void pyckbd_ ( )

8.12.2.2 void pyexec_ ( )

8.12.2.3 void pygive_ (const char * , int)

8.12.2.4 void pyjoin_ (int & , int &)

8.12.2.5 void pylist_ (int &)

8.12.2.6 double pymass_ (int &)

8.12.2.7 void pyname_ (int & , char * , int)

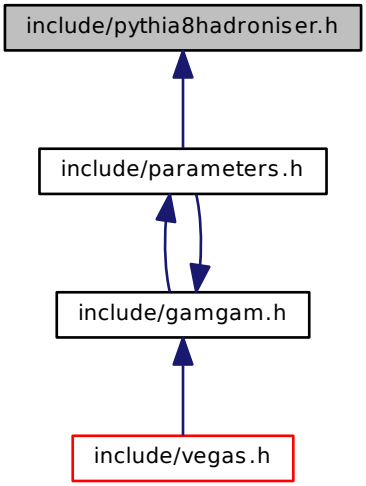
8.12.2.8 double pyp_ (int & , int &)

8.12.3 Variable Documentation

8.12.3.1 struct { ... } pyjets_

8.13 include/pythia8hadroniser.h File Reference

This graph shows which files directly or indirectly include this file:

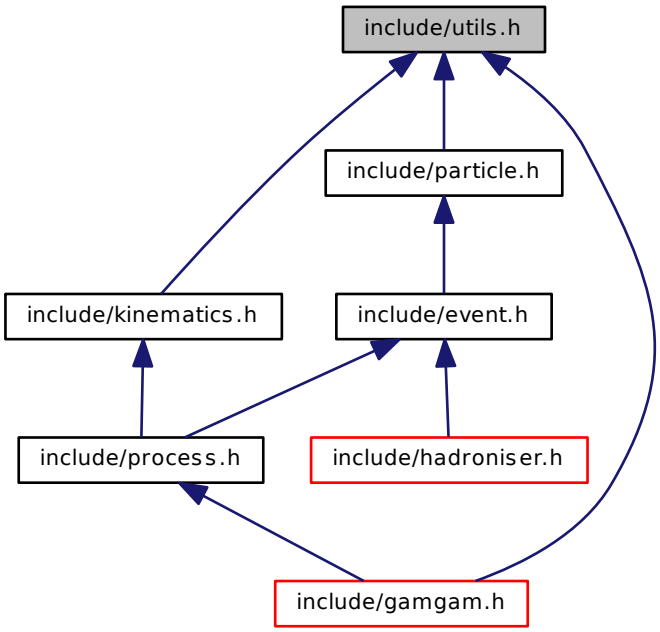


Data Structures

- class Pythia8Hadroniser

8.14 include/utils.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define alphaF 1./137.04

  Electromagnetic coupling constant $\alpha_{em} = \frac{e^2}{4\pi\epsilon_0\hbar c}$.
- #define muBarn 1./389.39

  $\frac{1}{(\hbar c)^2}$ $[\mathrm{b}^{-1}]$?
- #define pi 3.14159265358979323846264338
- #define sconst 3.89351824E8
- #define sconstb 2.1868465E10

Functions

- double GetMassFromPDGId (int pdgId_)

  Gets the mass of a particle.
- void Lorenb (double u_, double ps_[], double pi_[], double pf_[])
- void Map (double expo_, double xmin_, double xmax_, double *out_, double *dout_)

  Defines modified variables of integration to avoid peaks integrations (see [5] for details) Returns a set of two modified variables of integration to maintain the stability of the integrant. These two new variables are :
- void Mapla (double, double, int, double, double, double *, double *)
- bool PSF (double, double, double *, double *, double *)

8.14.1 Macro Definition Documentation

8.14.1.1 #define alphaF 1./137.04

8.14.1.2 #define muBarn 1./389.39

8.14.1.3 #define pi 3.14159265358979323846264338

8.14.1.4 #define sconst 3.89351824E8

8.14.1.5 #define sconstb 2.1868465E10

8.14.2 Function Documentation

8.14.2.1 double GetMassFromPDGId (int pdgId_)

Gets the mass in GeV/c**2 of a particle given its PDG identifier

Parameters

*pdgId_*	PDG ID of the particle whose mass is requested

Returns

Mass of the particle in GeV/c**2

8.14.2.2 void Lorenb (double u_, double ps_[], double pi_[], double pf_[])

8.14.2.3 void Map (double expo_, double xmin_, double xmax_, double * out_, double * dout_)

- $y_{out} = x_{min}\left(\frac{x_{max}}{x_{min}}\right)^{exp}$ the new variable
- $\mathrm{d}y_{out} = x_{min}\left(\frac{x_{max}}{x_{min}}\right)^{exp}\log\frac{x_{min}}{x_{max}}$, the new variable's differential form Redefines the variables of integration in order to avoid the strong peaking of the integrant.

  Parameters

*expo_*	Exponant
*xmin_*	Minimal value of the variable
*xmax_*	Maximal value of the variable
*out_*	The new variable definition
*dout_*	The differential variant of the new variable definition

  Note

  This method overrides the set of `mapxx` subroutines in ILPAIR, with a slight difference according to the sign of the $\mathrm{d}y_{out}$ parameter :
  - left unchanged :

    `mapw2`, `mapxq`, `mapwx`, `maps2`
  - opposite sign :

    `mapt1`, `mapt2`

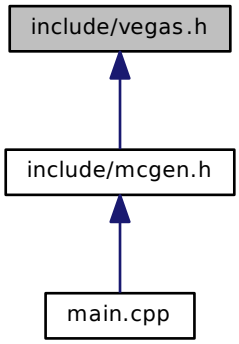
8.14.2.4 void Mapla (double , double , int , double , double , double * , double *)

8.14.2.5 bool PSF (double , double , double * , double * , double *)

Computes the proton structure function (F.W Brasse et al., DESY 76/11 (1976), http://dx.doi.org/10.1016/0550-3213(76)90231-5) [2]

8.15 include/vegas.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class Vegas

  Vegas Monte-Carlo integrator instance.

Macros

- #define MAX_ND 50

8.15.1 Macro Definition Documentation

8.15.1.1 #define MAX_ND 50

8.16 main.cpp File Reference

Functions

- int main (int argc, char *argv[])

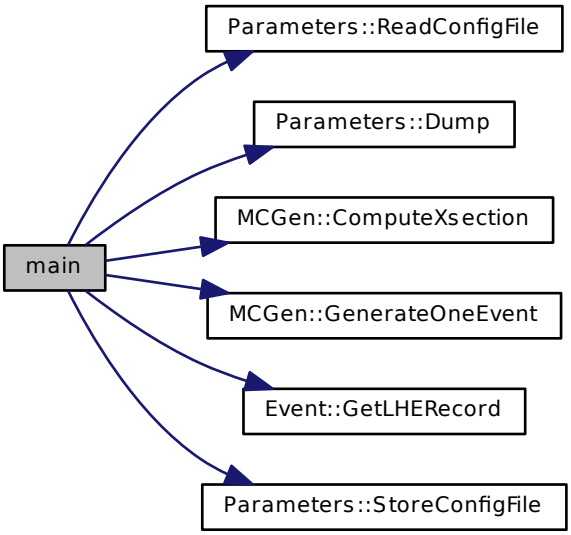
8.16.1 Function Documentation

8.16.1.1 int main (int argc, char * argv[])

Author

Laurent Forthomme laurent.forthomme@uclouvain.be Main caller for this Monte Carlo generator. Loads the configuration files' variables if set as an argument to this program, else loads a default "LHC-like" configuration, then launches the cross-section computation and the events generation.

Here is the call graph for this function:



References

[1] J. Beringer et al. Review of Particle Physics (RPP). *Phys.Rev.*, D86:010001, 2012. 41

[2] F.W. Brasse, W. Flauger, J. Gayler, S.P. Goel, R. Haidan, M. Merkwitz, and H. Wriedt. Parametrization of the q^2 dependence of $\gamma_\nu p$ total cross sections in the resonance region. *Nuclear Physics B*, 110(4–5):413 – 433, 1976. 65

[3] G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192 – 203, 1978. 2, 51

[4] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 0605:026, 2006. 48

[5] J.A.M. Vermaseren. Two-photon processes at very high energies. *Nuclear Physics B*, 229(2):347 – 371, 1983. 2, 11, 64

Index