LPAIR++

0.1

Generated by Doxygen 1.8.3.1

Fri Dec 13 2013 07:17:10

Contents

1 Todo List

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

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

4 Data Structure Documentation

4.1 Event Class Reference

Kinematic information on the particles in the event.

Public Member Functions

- void Dump ()
- Particle * GetByRole (int role_)
- int SetParticle (Particle *part_)
- void Store (std::ofstream *, double weight_=1.)
- void StoreLHERecord (std::ofstream *of_, const double weight_=1.)

  Stores the LHE block for this event.

4.1.1 Detailed Description

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

4.1.2 Member Function Documentation

4.1.2.1 void Event::Dump ( )

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

4.1.2.2 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

4.1.2.3 int Event::SetParticle (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

4.1.2.4 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

4.1.2.5 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

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

- include/event.h

4.2 GamGam Class Reference

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

Public Member Functions

- GamGam (const unsigned int ndim_, int nOpt_, double x_[])
  Class constructor.
- void ComputeSqS ()
  Computes $\sqrt{s}$ for the system.
- double ComputeXsec (int nm_=1)
  Computes the process' cross section.
- Particle * GetParticle (int role_)
  Get a particle given its role in the process.
- double GetT1 ()
- void GetT1extrema (double &t1min_, double &t1max_)
- double GetT2 ()
- void GetT2extrema (double &t2min_, double &t2max_)
- bool IsKinematicsDefined ()
  Is the system's kinematics well defined?
- void SetCuts (GamGamKinematics cuts_)
  Sets the list of kinematic cuts to apply on the outgoing particles' final state.
- bool SetIncomingKinematics (int part_, double momentum_[3], int pdgId_)
  Sets the momentum and PDG id for the incoming particles.
- bool SetIncomingKinematics (Particle ip1_, Particle ip2_)
  Sets the momentum and PDG id for the incoming particles.
- bool SetOutgoingParticles (int part_, int pdgId_)
  Sets the PDG id for the outgoing particles.

4.2.1 Detailed Description

Full class of methods and objects to compute the full analytic matrix element [2] 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).

4.2.2 Constructor & Destructor Documentation

4.2.2.1 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

4.2.3 Member Function Documentation

4.2.3.1 void GamGam::ComputeSqS ( )

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

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

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

Returns

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

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

Parameters

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

4.2.3.4 double GamGam::GetT1 ( ) [inline]

Returns the value for the first photon virtuality

Returns

t_1, the first photon virtuality

4.2.3.5 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

4.2.3.6 double GamGam::GetT2 ( ) [inline]

Returns the value for the second photon virtuality

Returns

t_2, the second photon virtuality

4.2.3.7 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

4.2.3.8 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

4.2.3.9 void GamGam::SetCuts (GamGamKinematics *cuts_*)

Parameters

*cuts_*	The Cuts object containing the kinematic parameters

4.2.3.10 bool GamGam::SetIncomingKinematics (int *part_*, double *momentum_[3]*, int *pdgId_*)

Specifies the incoming particles' kinematics as well as their properties (role in the process and code according to the PDG convention)

Parameters

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

Returns

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

4.2.3.11 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

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

Parameters

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

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

- include/gamgam.h

4.3 GamGamKinematics Class Reference

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

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 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.

4.3.1 Field Documentation

4.3.1.1 int GamGamKinematics::kinematics

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

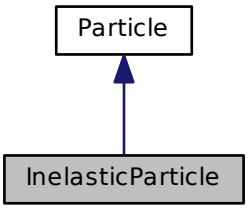
- 0 for the electron-electron elastic case
- 1 for the proton-proton elastic case
- 2 for the proton-proton single-dissociative (or inelastic) case
- 3 for the proton-proton double-dissociative case

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

- include/gamgam.h

4.4 InelasticParticle Class Reference

Inheritance diagram for InelasticParticle:



Public Member Functions

- void AddDaughter (Particle *part_)
  Specify a decay product for this particle.
- Particle * GetDaughter (const unsigned int num_=0)
  Gets a daughter from this particle, labelled by its identifier in this particle's daughters list.
- std::string GetLHEline (bool revert_=false)
- Particle * GetMother ()
  Gets the mother particle from which this particle arises.
- void Hadronise ()
  Hadronises the particle using Pythia.
- double M2 ()
  Gets the particle's squared mass.
- unsigned int NumDaughters ()
  Gets the number of daughter particles arising from this one.
- void SetE (double E_)
  Sets the particle's energy.
- void SetMother (Particle *part_)
  Sets the mother particle (from which this particle arises)
- bool SetP (double px_, double py_, double pz_)
  Sets the 3-momentum associated to the particle.
- bool SetP (double px_, double py_, double pz_, double E_)
  Sets the 4-momentum associated to the particle.
- bool SetP (double p_[3], double E_)
  Sets the 4-momentum associated to the particle.

Data Fields

- double e
  Energy, in GeV.
- double eta
  Pseudo-rapidity.
- bool isValid
  Is this particle a valid particle which can be used for kinematic computations ?

- double m
  Mass in GeV/c ^{2}.
- double p
  Norm of the 3-momentum, in GeV/c.
- int pdgId
  Particle Data Group integer identifier.
- double pt
  Transverse momentum, in GeV/c.
- 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.

4.4.1 Detailed Description

Class containing the information on a particle supposed to decay or fragment in the process

4.4.2 Member Function Documentation

4.4.2.1 void Particle::AddDaughter (Particle * *part_*) [inherited]

Parameters

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

4.4.2.2 Particle* Particle::GetDaughter (const unsigned int *num_* = 0) [inherited]

Returns

A Particle object containing all the kinematic information related to this daughter particle

4.4.2.3 std::string Particle::GetLHEline (bool *revert_* = false) [inherited]

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

4.4.2.4 void InelasticParticle::Hadronise ( )

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

4.4.2.5 void Particle::SetE (double *E_*) [inline], [inherited]

Parameters

*E_*	Energy, in GeV

4.4.2.6 void Particle::SetMother (Particle * *part_*) [inline], [inherited]

Parameters

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

4.4.2.7 bool Particle::SetP (double *px_*, double *py_*, double *pz_*) [inline], [inherited]

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

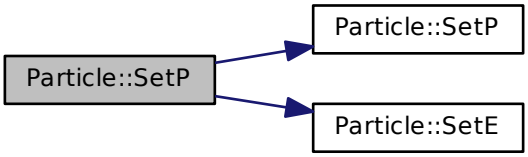
4.4.2.8 bool Particle::SetP (double *px_*, double *py_*, double *pz_*, double *E_*) [inline], [inherited]

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

Here is the call graph for this function:



4.4.2.9 bool Particle::SetP (double *p_[3]*, double *E_*) [inherited]

Parameters

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

4.4.3 Field Documentation

4.4.3.1 int Particle::status [inherited]

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

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

- include/inelastic.h

4.5 InputParameters Class Reference

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

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.
- 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 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 mintheta

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

  Number of events already generated in this run.
- 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.

4.5.1 Detailed Description

Note

The default parameters are derived from GMUINI in LPAIR

4.5.2 Member Function Documentation

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

Parameters

*inFile_*	Name of the configuration file to load

4.5.2.2 void InputParameters::SetEtaRange (double *etamin_*, double *etamax_*)

Parameters

*etamin_*	The minimal value of η\$ffortheoutgoingleptons@parametamax_Themaximalvalueof\$f for the outgoing leptons

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

Parameters

*outFile_*	Name of the configuration file to create

4.5.3 Field Documentation

4.5.3.1 bool InputParameters::debug

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

4.5.3.2 double InputParameters::maxmx

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

4.5.3.3 double InputParameters::maxpt

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

4.5.3.4 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

4.5.3.5 double InputParameters::minmx

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

4.5.3.6 double InputParameters::minpt

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

4.5.3.7 int InputParameters::ntreat

Note

Is it correctly implemented ?

4.5.3.8 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

4.5.3.9 int InputParameters::p2mod

Note

Was named EMOD in ILPAIR

4.5.3.10 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

4.5.3.11 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!

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

- include/utils.h

4.6 MCGen Class Reference

Core of the Monte-Carlo generator.

Public Member Functions

- MCGen (InputParameters ip_)
  Class constructor.
- void ComputeXsection (double *, double *)
- InputParameters GetInputParameters ()
  Returns the set of parameters used to setup the phase space to integrate.

4.6.1 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

4.6.2 Constructor & Destructor Documentation

4.6.2.1 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

4.6.3 Member Function Documentation

4.6.3.1 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.

4.6.3.2 InputParameters MCGen::GetInputParameters ( ) [inline]

Returns

The InputParameter object embedded in this class

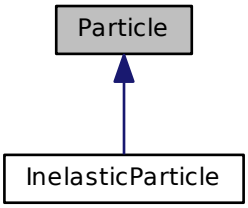
The documentation for this class was generated from the following file:

- include/mcgen.h

4.7 Particle Class Reference

Kinematics of one particle.

Inheritance diagram for Particle:



Public Member Functions

- void AddDaughter (Particle *part_)

  Specify a decay product for this particle.
- Particle * GetDaughter (const unsigned int num_=0)

  Gets a daughter from this particle, labelled by its identifier in this particle's daughters list.
- std::string GetLHEline (bool revert_=false)
- Particle * GetMother ()

Gets the mother particle from which this particle arises.

- double M2 ()

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

  Gets the number of daughter particles arising from this one.
- void SetE (double E_)

  Sets the particle's energy.
- void SetMother (Particle *part_)

  Sets the mother particle (from which this particle arises)
- bool SetP (double px_, double py_, double pz_)

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

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

  Sets the 4-momentum associated to the particle.

Data Fields

- double e

  Energy, in GeV.
- double eta

  Pseudo-rapidity.
- bool isValid

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

  Mass in GeV/c ^{2}.
- double p

  Norm of the 3-momentum, in GeV/c.
- int pdgId

  Particle Data Group integer identifier.
- double pt

  Transverse momentum, in GeV/c.
- 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.

4.7.1 Detailed Description

Kinematic information for one particle

4.7.2 Member Function Documentation

4.7.2.1 void Particle::AddDaughter (Particle * *part_*)

Parameters

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

4.7.2.2 Particle* Particle::GetDaughter (const unsigned int *num_* = 0)

Returns

A Particle object containing all the kinematic information related to this daughter particle

4.7.2.3 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

4.7.2.4 void Particle::SetE (double *E_*) [inline]

Parameters

*E_*	Energy, in GeV

4.7.2.5 void Particle::SetMother (Particle * *part_*) [inline]

Parameters

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

4.7.2.6 bool Particle::SetP (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

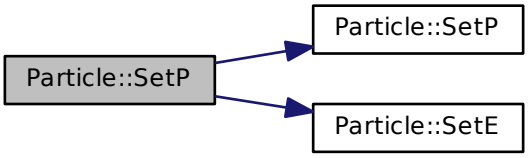
4.7.2.7 bool Particle::SetP (double *px_*, double *py_*, double *pz_*, double *E_*) [inline]

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

Here is the call graph for this function:



4.7.2.8 bool Particle::SetP (double *p_[3]*, double *E_*)

Parameters

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

4.7.3 Field Documentation

4.7.3.1 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

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

- include/particle.h

4.8 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.
- 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_)

4.8.1 Constructor & Destructor Documentation

4.8.1.1 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. [1]

Parameters

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

4.8.2 Member Function Documentation

4.8.2.1 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

4.8.2.2 int Vegas::LaunchGeneration ( )

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

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

Vegas algorithm to perform the (_dim)-dimensional Monte Carlo integration of a given function as described in[1]

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

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

- include/vegas.h

References

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

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

Index