

LPAIR++
0.1

Generated by Doxygen 1.8.3.1

Wed Aug 7 2013 13:58:05

Contents

1	Todo List	1
2	Data Structure Index	2
2.1	Data Structures	2
3	Data Structure Documentation	2
3.1	Cuts Class Reference	2
3.1.1	Constructor & Destructor Documentation	3
3.1.2	Field Documentation	3
3.2	GamGam Class Reference	3
3.2.1	Constructor & Destructor Documentation	4
3.2.2	Member Function Documentation	4
3.3	InelasticParticle Class Reference	6
3.3.1	Constructor & Destructor Documentation	6
3.3.2	Member Function Documentation	6
3.4	InputParameters Class Reference	6
3.4.1	Detailed Description	8
3.4.2	Constructor & Destructor Documentation	8
3.4.3	Member Function Documentation	8
3.4.4	Field Documentation	8
3.5	MCGen Class Reference	10
3.5.1	Detailed Description	10
3.5.2	Constructor & Destructor Documentation	10
3.5.3	Member Function Documentation	10
3.6	Particle Class Reference	11
3.6.1	Detailed Description	12
3.6.2	Constructor & Destructor Documentation	12
3.6.3	Member Function Documentation	12
3.6.4	Field Documentation	13
3.7	Vegas Class Reference	14
3.7.1	Constructor & Destructor Documentation	14
3.7.2	Member Function Documentation	14

Index	15
--------------	-----------

1 Todo List

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

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

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

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Cuts	
List of kinematic cuts to apply on the central and outgoing phase space	2
GamGam	
Computes the matrix element for a $\gamma\gamma \rightarrow \ell^+\ell^-$ process	3
InelasticParticle	6
InputParameters	
List of input parameters used to start and run the simulation job	6
MCGen	
Core of the Monte-Carlo generator ; Computes the cross section for any value of the input parameters by calling Vegas on GamGam objects	10
Particle	
Kinematics of one particle	11
Vegas	
Vegas Monte-Carlo integrator instance	14

3 Data Structure Documentation

3.1 Cuts Class Reference

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

Public Member Functions

- [Cuts](#) ()
- [~Cuts](#) ()

Data Fields

- double [emax](#)
 Maximal energy of the central two-photons system.
- double [emin](#)
 Minimal energy of the central two-photons system.
- int [mode](#)
 Sets of cuts to apply on the final phase space.
- double [mxmax](#)
- double [mxmin](#)
- double [ptmax](#)
 Maximal transverse momentum of the single outgoing leptons.

- double [ptmin](#)
Minimal transverse momentum of the single outgoing leptons.
- double [thetamax](#)
Maximal polar (θ_{\max}) angle of the outgoing leptons, expressed in degrees.
- double [thetamin](#)
Minimal polar (θ_{\min}) angle of the outgoing leptons, expressed in degrees.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 [Cuts::Cuts \(\)](#)

3.1.1.2 [Cuts::~~Cuts \(\)](#)

3.1.2 Field Documentation

3.1.2.1 double [Cuts::emax](#)

3.1.2.2 double [Cuts::emin](#)

3.1.2.3 int [Cuts::mode](#)

3.1.2.4 double [Cuts::mxmax](#)

3.1.2.5 double [Cuts::mxmin](#)

3.1.2.6 double [Cuts::ptmax](#)

3.1.2.7 double [Cuts::ptmin](#)

3.1.2.8 double [Cuts::thetamax](#)

3.1.2.9 double [Cuts::thetamin](#)

3.2 GamGam Class Reference

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

Public Member Functions

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

- bool [SetIncomingKinematics](#) ([Particle](#), [Particle](#))
Sets the momentum and PDG id for the incoming particles.
- bool [SetOutgoingParticles](#) (int, int)
Sets the PDG id for the outgoing particles.
- void [SetWRange](#) (double, double)
Sets the energy range available for the phase space integration.
- void [StoreEvent](#) (std::ofstream *, double)

3.2.1 Constructor & Destructor Documentation

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

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

Parameters

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

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

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

3.2.1.2 GamGam::~~GamGam ()

3.2.2 Member Function Documentation

3.2.2.1 void GamGam::ComputeSqS ()

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

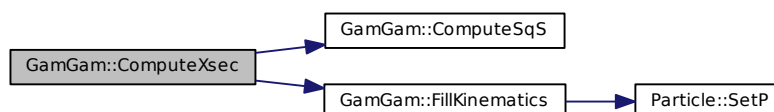
3.2.2.2 double GamGam::ComputeXsec (int *nm_* = 1)

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

Returns

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

Here is the call graph for this function:



3.2.2.3 void GamGam::FillKinematics ()

Here is the call graph for this function:

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

Parameters

<i>role_</i>	An integer denoting the particle's role in the selected production process
--------------	--

3.2.2.5 bool GamGam::IsKinematicsDefined () [inline]

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

Returns

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

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

Parameters

<i>cuts_</i>	The Cuts object containing the kinematic parameters
--------------	---

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

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

Parameters

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

Returns

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

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

Specifies the incoming particles' kinematics as well as their properties using two [Particle](#) objects.

Parameters

<i>ip1_</i>	Information on the first incoming particle
<i>ip2_</i>	Information on the second incoming particle

3.2.2.9 bool GamGam::SetOutgoingParticles (int *part_*, int *pdgld_*)

Parameters

<i>part_</i>	Role of the particle in the process
<i>pdgld_</i>	Particle ID according to the PDG convention

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

Parameters

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

3.2.2.11 void GamGam::StoreEvent (std::ofstream * *file_*, double *weight_* = 1.)

Here is the call graph for this function:



3.3 InelasticParticle Class Reference

Public Member Functions

- [InelasticParticle](#) ()
- [~InelasticParticle](#) ()
- void [Hadronise](#) ()
- void [PDF2PDG](#) ()

3.3.1 Constructor & Destructor Documentation

3.3.1.1 InelasticParticle::InelasticParticle ()

3.3.1.2 InelasticParticle::~~InelasticParticle ()

3.3.2 Member Function Documentation

3.3.2.1 void InelasticParticle::Hadronise ()

3.3.2.2 void InelasticParticle::PDF2PDG ()

3.4 InputParameters Class Reference

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

Public Member Functions

- [InputParameters](#) ()

- `~InputParameters ()`
- `void Dump ()`
Dumps the input parameters in the console.
- `bool ReadConfigFile (std::string)`
Reads content from config file to load the variables.
- `bool StoreConfigFile (std::string)`
Stores the full run configuration to an external config file.

Data Fields

- `bool debug`
Do we need control plots all along the process?
- `std::ofstream * file`
The file in which to store the events generation's output.
- `std::ofstream * file_debug`
- `bool generation`
Are we generating events ? (true) or are we only computing the cross-section ? (false)
- `double in1p`
First incoming particle's momentum (in GeV/c)
- `double in2p`
Second incoming particle's momentum (in GeV/c)
- `int itmx`
Maximal number of iterations to perform by VEGAS.
- `int itvg`
Number of Vegas integrations.
- `double maxenergy`
Maximal energy of the outgoing leptons.
- `double maxmx`
- `double maxpt`
Maximal transverse momentum 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`
- `double minpt`
Minimal transverse momentum of the outgoing leptons.
- `double mintheta`
Minimal polar angle θ of the outgoing leptons.
- `int ncvg`
- `int ngen`
Number of events already generated in this run.
- `int p1mod`
First particle's mode.
- `int p2mod`
Second particle's mode.
- `int pair`
PDG id of the outgoing leptons.
- `Gnuplot * plot [MAX_HISTOS]`
Control plots objects.
- `bool store`
Are the events generated in this run to be stored in the output file ?

3.4.1 Detailed Description

Note

The default parameters are derived from GMUINI in LPAIR

3.4.2 Constructor & Destructor Documentation

3.4.2.1 InputParameters::InputParameters ()

3.4.2.2 InputParameters::~~InputParameters ()

3.4.3 Member Function Documentation

3.4.3.1 void InputParameters::Dump ()

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

Parameters

<i>inFile_</i>	Name of the configuration file to load
----------------	--

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

Parameters

<i>outFile_</i>	Name of the configuration file to create
-----------------	--

3.4.4 Field Documentation

3.4.4.1 bool InputParameters::debug

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

3.4.4.2 std::ofstream* InputParameters::file

3.4.4.3 std::ofstream* InputParameters::file_debug

3.4.4.4 bool InputParameters::generation

3.4.4.5 double InputParameters::in1p

3.4.4.6 double InputParameters::in2p

3.4.4.7 int InputParameters::itmx

3.4.4.8 int InputParameters::itvg

3.4.4.9 double InputParameters::maxenergy

3.4.4.10 double InputParameters::maxmx

3.4.4.11 double InputParameters::maxpt

3.4.4.12 double InputParameters::maxtheta

3.4.4.13 int InputParameters::mcut

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

- 0 - No cuts at all (for the total cross section)

- 1 - Vermaserens' hypothetical detector cuts : for both leptons,
 - $\frac{|p_z|}{|\mathbf{p}|} \leq 0.75$ and $p_T \geq 1$ GeV, or
 - $0.75 < \frac{|p_z|}{|\mathbf{p}|} \leq 0.95$ and $p_z > 1$ GeV,
- 2 - [Cuts](#) according to the provided parameters

3.4.4.14 `double InputParameters::minenergy`

3.4.4.15 `double InputParameters::minmx`

3.4.4.16 `double InputParameters::minpt`

3.4.4.17 `double InputParameters::mintheta`

3.4.4.18 `int InputParameters::ncvg`

3.4.4.19 `int InputParameters::ngen`

3.4.4.20 `int InputParameters::p1mod`

The first incoming particle type and kind of interaction :

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

Note

Was named PMOD in ILPAIR

3.4.4.21 `int InputParameters::p2mod`

Note

Was named EMOD in ILPAIR

3.4.4.22 `int InputParameters::pair`

The particle code of produced leptons :

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

3.4.4.23 `Gnuplot* InputParameters::plot[MAX_HISTOS]`

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

Note

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

3.4.4.24 bool InputParameters::store

3.5 MCGen Class Reference

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

Public Member Functions

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

3.5.1 Detailed Description

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

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

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

Author

Laurent Forthomme laurent.forthomme@uclouvain.be

Date

February 2013

3.5.2 Constructor & Destructor Documentation

3.5.2.1 MCGen::MCGen (InputParameters ip_)

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

Parameters

<i>ip_</i>	List of input parameters defining the phase space on which to perform the integration
------------	---

3.5.2.2 MCGen::~~MCGen ()

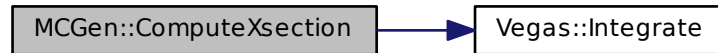
3.5.3 Member Function Documentation

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

3.5.3.2 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.

Here is the call graph for this function:



3.5.3.3 InputParameters MCGen::GetInputParameters () [inline]

Returns

The InputParameter object embedded in this class

3.5.3.4 void MCGen::LaunchGen (const unsigned int *count*_)

Here is the call graph for this function:



3.5.3.5 void MCGen::Test ()

Here is the call graph for this function:



3.6 Particle Class Reference

Kinematics of one particle.

Public Member Functions

- [Particle](#) ()
- [~Particle](#) ()
- `std::string` [GetLHEline](#) (bool revert__{__}=false)
- void [SetE](#) (double E__{__})
- void [SetP](#) (double px__{__}, double py__{__}, double pz__{__})
Sets the 3-momentum associated to the particle.
- void [SetP](#) (double px__{__}, double py__{__}, double pz__{__}, double E__{__})
Sets the 4-momentum associated to the particle.

Data Fields

- double [e](#)
Energy in GeV.
- double [m](#)
Mass in GeV/c^2 .
- double [p](#)
Momentum.
- int [pdgId](#)
Particle Data Group integer identifier.
- double [pt](#)
Transverse momentum.
- double [px](#)
Momentum along the x-axis in GeV/c .
- double [py](#)
Momentum along the y-axis in GeV/c .
- double [pz](#)
Momentum along the z-axis in GeV/c .
- int [role](#)
Role in the considered process.

3.6.1 Detailed Description

Kinematic information for one particle

3.6.2 Constructor & Destructor Documentation

3.6.2.1 `Particle::Particle ()`3.6.2.2 `Particle::~~Particle ()`

3.6.3 Member Function Documentation

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

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

Parameters

<code>revert_</code>	Is the event symmetric ? If set to true, the third component of the momentum is reverted.
----------------------	---

Returns

The LHE line

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

3.6.3.3 void Particle::SetP (double *px_*, double *py_*, double *pz_*) [inline]

Parameters

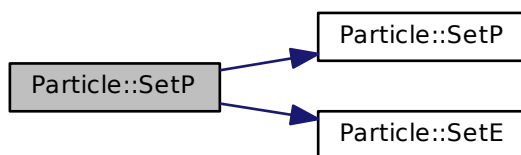
<i>px_</i>	Momentum along the <i>x</i> -axis
<i>py_</i>	Momentum along the <i>y</i> -axis
<i>pz_</i>	Momentum along the <i>z</i> -axis

3.6.3.4 void Particle::SetP (double *px_*, double *py_*, double *pz_*, double *E_*) [inline]

Parameters

<i>px_</i>	Momentum along the <i>x</i> -axis
<i>py_</i>	Momentum along the <i>y</i> -axis
<i>pz_</i>	Momentum along the <i>z</i> -axis
<i>E_</i>	Energy

Here is the call graph for this function:



3.6.4 Field Documentation

3.6.4.1 double Particle::e

3.6.4.2 double Particle::m

3.6.4.3 double Particle::p

3.6.4.4 int Particle::pdgId

3.6.4.5 double Particle::pt

3.6.4.6 double Particle::px

3.6.4.7 double Particle::py

3.6.4.8 double Particle::pz

3.6.4.9 int Particle::role

3.7 Vegas Class Reference

[Vegas](#) Monte-Carlo integrator instance.

Public Member Functions

- [Vegas](#) (int, double f__(double *, size_t, void *), [InputParameters](#) *inParam_)
- [~Vegas](#) ()
Class destructor.
- int [Generate](#) (int)
- int [Integrate](#) (double *, double *)
Launches the integration of the provided function.
- int [LaunchGeneration](#) (int)
Launches the generation of events.
- void [SetGen](#) ()

3.7.1 Constructor & Destructor Documentation

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

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

Parameters

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

3.7.1.2 Vegas::~~Vegas ()

3.7.2 Member Function Documentation

3.7.2.1 int Vegas::Generate (int)

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

Launches the [Vegas](#) integration of the provided function with the provided input parameters.

Parameters

<i>result_</i>	The cross section as integrated by Vegas for the given phase space restrictions
<i>abserr_</i>	The error associated to the computed cross section

3.7.2.3 int Vegas::LaunchGeneration (int nEvts_)

Launches the [Vegas](#) generation of events according to the provided input parameters.

Parameters

<i>nEvts_</i>	The number of events to generate
---------------	----------------------------------

3.7.2.4 void Vegas::SetGen ()

References

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

Index

- ~Cuts
 - Cuts, [2](#)
- ~GamGam
 - GamGam, [3](#)
- ~InelasticParticle
 - InelasticParticle, [6](#)
- ~InputParameters
 - InputParameters, [7](#)
- ~MCGen
 - MCGen, [10](#)
- ~Particle
 - Particle, [12](#)
- ~Vegas
 - Vegas, [14](#)
- AnalyzePhaseSpace
 - MCGen, [10](#)
- ComputeSqS
 - GamGam, [3](#)
- ComputeXsec
 - GamGam, [3](#)
- ComputeXsection
 - MCGen, [10](#)
- Cuts, [1](#)
 - ~Cuts, [2](#)
 - Cuts, [2](#)
 - emax, [2](#)
 - emin, [2](#)
 - mode, [2](#)
 - mxmax, [2](#)
 - mxmin, [2](#)
 - ptmax, [2](#)
 - ptmin, [2](#)
 - thetamax, [2](#)
 - thetamin, [2](#)
- debug
 - InputParameters, [7](#)
- Dump
 - InputParameters, [7](#)
- e
 - Particle, [13](#)
- emax
 - Cuts, [2](#)
- emin
 - Cuts, [2](#)
- file
 - InputParameters, [7](#)
- file_debug
 - InputParameters, [8](#)
- FillKinematics
 - GamGam, [4](#)
- GamGam, [2](#)
- ~GamGam, [3](#)
- ComputeSqS, [3](#)
- ComputeXsec, [3](#)
- FillKinematics, [4](#)
- GamGam, [3](#)
- GamGam, [3](#)
- GetParticle, [4](#)
- IsKinematicsDefined, [4](#)
- SetCuts, [4](#)
- SetIncomingKinematics, [4](#), [5](#)
- SetOutgoingParticles, [5](#)
- SetWRange, [5](#)
- StoreEvent, [5](#)
- Generate
 - Vegas, [14](#)
- generation
 - InputParameters, [8](#)
- GetInputParameters
 - MCGen, [10](#)
- GetLHEline
 - Particle, [12](#)
- GetParticle
 - GamGam, [4](#)
- Hadronise
 - InelasticParticle, [6](#)
- in1p
 - InputParameters, [8](#)
- in2p
 - InputParameters, [8](#)
- InelasticParticle, [5](#)
 - ~InelasticParticle, [6](#)
 - Hadronise, [6](#)
 - InelasticParticle, [6](#)
 - InelasticParticle, [6](#)
 - PDF2PDG, [6](#)
- InputParameters, [6](#)
 - ~InputParameters, [7](#)
 - debug, [7](#)
 - Dump, [7](#)
 - file, [7](#)
 - file_debug, [8](#)
 - generation, [8](#)
 - in1p, [8](#)
 - in2p, [8](#)
 - InputParameters, [7](#)
 - InputParameters, [7](#)
 - itmx, [8](#)
 - itvg, [8](#)
 - maxenergy, [8](#)
 - maxmx, [8](#)
 - maxpt, [8](#)
 - maxtheta, [8](#)
 - mcut, [8](#)
 - minenergy, [8](#)

- minmx, 8
- minpt, 8
- mintheta, 8
- ncvg, 8
- ngen, 8
- p1mod, 8
- p2mod, 8
- pair, 9
- plot, 9
- ReadConfigFile, 7
- store, 9
- StoreConfigFile, 7
- Integrate
 - Vegas, 14
- IsKinematicsDefined
 - GamGam, 4
- itmx
 - InputParameters, 8
- itvg
 - InputParameters, 8
- LaunchGen
 - MCGen, 11
- LaunchGeneration
 - Vegas, 14
- m
 - Particle, 13
- MCGen, 9
 - ~MCGen, 10
 - AnalyzePhaseSpace, 10
 - ComputeXsection, 10
 - GetInputParameters, 10
 - LaunchGen, 11
 - MCGen, 10
 - MCGen, 10
 - Test, 11
- maxenergy
 - InputParameters, 8
- maxmx
 - InputParameters, 8
- maxpt
 - InputParameters, 8
- maxtheta
 - InputParameters, 8
- mcut
 - InputParameters, 8
- minenergy
 - InputParameters, 8
- minmx
 - InputParameters, 8
- minpt
 - InputParameters, 8
- mintheta
 - InputParameters, 8
- mode
 - Cuts, 2
- mxmax
 - Cuts, 2
- mxmin
 - Cuts, 2
- ncvg
 - InputParameters, 8
- ngen
 - InputParameters, 8
- p
 - Particle, 13
- p1mod
 - InputParameters, 8
- p2mod
 - InputParameters, 8
- PDF2PDG
 - InelasticParticle, 6
- pair
 - InputParameters, 9
- Particle, 11
 - ~Particle, 12
 - e, 13
 - GetLHEline, 12
 - m, 13
 - p, 13
 - Particle, 12
 - pdgId, 13
 - pt, 13
 - px, 13
 - py, 13
 - pz, 13
 - role, 13
 - SetE, 12
 - SetP, 12
- pdgId
 - Particle, 13
- plot
 - InputParameters, 9
- pt
 - Particle, 13
- ptmax
 - Cuts, 2
- ptmin
 - Cuts, 2
- px
 - Particle, 13
- py
 - Particle, 13
- pz
 - Particle, 13
- ReadConfigFile
 - InputParameters, 7
- role
 - Particle, 13
- SetCuts
 - GamGam, 4
- SetE
 - Particle, 12

- SetGen
 - Vegas, [14](#)
- SetIncomingKinematics
 - GamGam, [4](#), [5](#)
- SetOutgoingParticles
 - GamGam, [5](#)
- SetP
 - Particle, [12](#)
- SetWRange
 - GamGam, [5](#)
- store
 - InputParameters, [9](#)
- StoreConfigFile
 - InputParameters, [7](#)
- StoreEvent
 - GamGam, [5](#)
- Test
 - MCGen, [11](#)
- thetamax
 - Cuts, [2](#)
- thetamin
 - Cuts, [2](#)
- Vegas, [13](#)
 - ~Vegas, [14](#)
 - Generate, [14](#)
 - Integrate, [14](#)
 - LaunchGeneration, [14](#)
 - SetGen, [14](#)
 - Vegas, [13](#)