Contents

**Basic Information**

[Overview**:**3](#_Toc461369149)

[Installation:4](#_Toc461369150)

[Input:6](#_Toc461369151)

[DPMJET:13](#_Toc461369152)

[Output15](#_Toc461369153)

[Class Diagram17](#_Toc461369154)

[File Descriptions18](#_Toc461369155)

Recent Updates:

v311: Added channel: ω -> π+π-π0 with code (223211111).

v309: Added channels: ρ0 -> e+e- and ρ0 -> μ+μ- with codes(113011) and (113013) respectively.

v308: Added an option to display a header in the output file. This header will show input parameters in a similar format to eSTARlight. Option is controlled by input parameter "OUTPUT\_HEADER", see below.

v307: Fixed 4-prong mass spectrum, properly converting dsigma/ds to dsigma/dW. The net effect is to scale the mass spectrum by 1/M\_{4\pi}, reducing the number of high-mass states.

v306: Updated gammavm.cpp, to properly output 4 pions with net charge 0. Also changed default Wmax for 4-prong final state to be the larger of the kinematic limit or 10 GeV; previously, it was unduly large for the LHC.

v305: Changed coefficient in calculation of pt2 in gammavm.cpp from 8 to 32. This changes the maximum p\_T for vector mesons for calculations without interference from about 250 MeV/c to about 1 Gev/c. In the long run, we could rename INT\_PT\_MAX and make it required parameter with or without interference. This could make the running a bit more efficient.

v304: Fixed a bug in gammaaluminosity.cpp lines 404, where photonDensity was called with its arguments reversed. This bug affected the p\_T spectrum when interference is turned on.

v299: Added hard-coded Woods-Saxon radii, thickness and density for 96Ru and 96Zr, for the RHIC isobar run. Data is from arXiv:1607.04697

v297: Changed normalized for Woods-Saxon density for non-predefined (i. e. not gold, lead, xenon or copper or nuclei with Z<7) so that the density is properly normalized \int d^3r rho(r ) = A. The normalization was previously done for a hard-sphere nucleus, so this over-estimated the cross-sections by 5-10%.

v295: Added hard-coded values for xenon-129 to match the recent LHC run. Radius=5.36 fm, density=0.18406

v293: Introduced shared random number generator which can be externally passed by the user. All particle constants (masses, widths, branching ratios, and spins) can now also be set by the user, but should be changed from the default values with care.

v290: Added an new BREAKUP\_MODE option to generate two-photon events in peripheral collisions. BREAKUP\_MODE=8 sets a fixed impact parameter range, regardless of the presence of nuclear breakup; it is intended to study two-photon production in peripheral collisions. It requires two additional otherwise optional input lines, BMIN and BMAX, to set the impact parameter range. It does not (yet?) work for photonuclear interactions.

278: Added two new optional parameters:

IMPULSE\_VM Normally 0, but can be set to 1 to perform an impulse approximation calculation (i.e. ignoring nuclear effects)

QUANTUM\_GLAUBER. When set to 1, performs a quantum Glauber calculation, rather than a classical one. This leads to greatly increased rho and omega cross-sections for heavy nuclei, little effect for heavier mesons.

Also added a final state, 4432212, for J/psi -> pbar p

v276: Added two new optional parameters (BSLOPE DEFINITION and BSLOPE\_VALUE) for the pT spectrum ('bslope') for proton targets or incoherent production on nuclei

v275: Added gg to axion channel as two-photon channel 88, per S. Knapen et al., arXiv:1607.07083 v273: "Baseline" version, described in arXiv:1607.03838)

Overview**:**

The STARlight Monte Carlo models 2-photon and photon-Pomeron interactions in ultra-peripheral heavy ion collisions. The physics approach for the photon-Pomeron interactions is described in Klein and Nystrand, Phys. Rev. C60, 014903 (1999), with the p\_t spectrum (including vector meson interference) discussed in Phys. Rev. Lett. 84, 2330 (2000). The 2-photon interactions are described in Baltz, Gorbunov, Klein, Nystrand, Phys.Rev. C80 044902 (2009).

STARlight has several input files, all of which are expected to be in the same directory as the starlight code. User-specified input parameters are read from a file named "slight.in"; these parameters are described below in [Input](#Input).

The simulated events are written to an ASCII file named "slight.out", which is described below in [Output](#Output).

Installation:

To install & run STARlight in a \*nix based environment, follow these steps(README):

Download the starlight package from 'Downloads' on the left sidebar of the homepage. The version in the example might be outdated.

-wget '<https://starlight.hepforge.org/downloads?f=starlight_r300.tar>'

-mv 'downloads?f=starlight\_r300.tar' starlight\_r300.tar

-tar xvf starlight\_r300.tar

Alternatively, one may obtain the latest version via svn.

HEPforge uses phabricator and no longer allows for anonymous checkouts of the repository.  (Please read <https://www.hepforge.org/guide.pdf> .)

To obtain an account, register here:  <https://www.hepforge.org/register>

Once you are registered, login:

<https://phab.hepforge.org/auth/start/?next=%2F>

Set up a version control settings (VCS) password under your account's Settings->AUTHENTICATION->VCS password .  The VCS password is needed to checkout the code.

(For remote users) To identify yourself, upload a SSH public key under account's Settings->AUTHENTICATION->SSH Public Keys with the button SSH Key Actions->Upload Public Key. This key will provide your identity when checking out the code as VCS. If you do not have a public ssh key to upload, you may generate a pair on the same SSH Public Keys page with the button SSH Key Actions-> Generate Keypair.

With the private ssh key loaded on your machine (and public on their machine), use svn to checkout the trunk/:

-svn co svn+ssh://[vcs@phab.hepforge.org](mailto:vcs@phab.hepforge.org)/source/starlightsvn/trunk

Change to the installation directory of your choice

-mkdir /home/my/installation/dir

-cd /home/my/installation/dir

Setup the compilation with cmake

-cmake /path/to/trunk

Compile with (g)make

-gmake

Setup the input file, slight.in, for your simluation needs

-cp /path/to/trunk/config/slight.in .

-vim slight.in

Run

-./starlight >& output.txt&

For more information and special scenarios, such as running with PYTHIA or DPMJET, consult the README files located in trunk/

If you would like to browse the code, please visit: <https://phab.hepforge.org/source/starlightsvn/>

------Before HEPForge updated their repository management system----

To obtain the latest version:

-svn co <http://starlight.hepforge.org/svn/trunk>

Alternatively:

-Visit <https://starlight.hepforge.org/trac/browser>

-Download the trunk [click on the download symbol in the Size column]

-Unpackage the zip file. The trunk/ represents <PathToSource>

To build Starlight:

- First create your build directory <BUILDDIR> (e.g. mkdir bin)

- $ cd <BUILDDIR>

- $ cmake <PathToSource>

- $ make

This creates an executable file, starlight, in the build directory.

To clean the build:

- $ make clean

To run starlight, a configuration file, slight.in, is needed. Examples of

slight.in may be found in the config/ directory.

To run:

$ ./starlight

Enabling Pythia:

To simulate the , ’, and c channels, you need Pythia v8.2 or higher to handle their decays. To enable Pythia support you need to run cmake with the option –DENABLE\_PYTHIA=ON and have $PYTHIADIR pointing to the top directory of Pythia8. [Note: when building Pythia, be sure to enable shared libraries(.so). ./configure –-enable-shared before compiling Pythia.]

$ setenv PYTHIADIR /my/local/pythia8

$ cmake <PathToSource> -DENABLE\_PYTHIA=ON

Note: v8.2+ is necessary since the Pythia directory structure changed[trunk/cmake\_modules/FindPythia8.cmake depends on the structure layout], liblhapdfdummy was removed, and Standalone:allowResDec was removed.

To enable DPMJET, please see the passage on [DPMJET](#DPMJET)

Input:

The input parameters are listed below with typical values for LHC Pb-Pb running given in parentheses. Optional parameters are denoted with \*.

baseFileName # The name of the output files. STARlight will copy the input slight.in to baseFileName.in, and produce output files baseFileName.txt and baseFileName.out. (slight)

BEAM\_1\_Z = 82 # Charge of beam one projectile. (82)

BEAM\_1\_A = 208 # Atomic number of beam one projectile. (208)

BEAM\_2\_Z = 82 # Charge of beam two projectile. (82)

BEAM\_2\_A = 208 # Atomic number of beam two projectile. (208)

BEAM\_1\_GAMMA = 1470 # Lorentz boost for beam one projectile(pz>0). (1470)

BEAM\_2\_GAMMA = 1470.0 # Lorentz boost for beam two projectile(pz<0). (1470)

W\_MAX = 12.0 # Maximum value for the gamma-gamma center of mass energy, W = 4E1E2, in GeV. Setting W\_MAX = -1 tells STARlight to use the default value specified in inputParameters.cpp (recommended for single meson production). For single mesons, the default W\_MAX is the particle mass plus five times the width. For lepton pairs, the default W\_MAX is given by 2. These are defined in src/inputParameters.cpp (-1)

W\_MIN = -1 #Min value of w. Minimum value for the gamma-gamma center of mass energy, W = 4E1E2, in GeV. Setting W\_MIN = -1 tells STARlight to use the default value specified in inputParameters.cpp (recommended for single meson production). The default W\_MIN is the larger of the kinematic limit ( *e.g.* 2m for  decays) or the particle mass minus five times the width. (-1)

W\_N\_BINS = 40 #Bins w maximum and minimum values for w (the gamma-gamma center of mass energy, w = 4E1E2), and the number of w bins in the lookup tables (40)

RAP\_MAX = 8. # Maximum rapidity of produced particle. (8)

RAP\_N\_BINS = 80 # Number of rapidity bins used in the cross section calculation (80)

CUT\_PT\* = 0 # Specifies whether the user chooses to place restrictions on the transverse momentum of the decay products. 0= no, 1 = yes. (0)

PT\_MIN\* = 1.0 # If a transverse momentum cut is applied, this specifies the minimum value produced, in GeV/c. (1.0)

PT\_MAX\* = 3.0 # If a transverse momentum cut is applied, this specifies the maximum value produced, in GeV/c. (3.0)

CUT\_ETA\* = 0 # Specifies whether the user chooses to place restrictions on the pseudorapidity of the decay products. 0= no, 1 = yes. (0)

ETA\_MIN\* = -10 # If a pseudorapidity cut is applied, this specifies the minimum value produced. (-10)

ETA\_MAX\* = 10 # If a pseudorapidity cut is applied, this specifies the maximum value produced. (10)

PROD\_MODE = 2 #**PROD\_MODE=1**: Two-photon interaction. **PROD\_MODE=2**: Coherent photonuclear vector meson production assuming narrow resonances. This option should also be used for exclusive vector meson production in pp collision. In pA or pp collisions, this option means that the proton emits the photon and that the gamma-A interaction is coherent.

**PROD\_MODE=3**: Coherent photonuclear vector meson production assuming wide resonances. This option should in be used for exclusive $\rho^0$ production.

**PROD\_MODE=4**: Incoherent photonuclear vector meson production. In pA collisions, this option means that the nucleus emits the photon. Do not use for pp.

**PROD\_MODE=5**: Photonuclear one photon exchange uses DPMJET single.

**PROD\_MODE=6**: Photonuclear two photon exchange (both nuclei excited) uses DPMJET double.

**PROD\_MODE=7**: Photonuclearsinglepa uses DPMJET Single, proton mode.

**PROD\_MODE=8**: [not supported/verified] Photonuclear singlepapy uses Pythia 6

N\_EVENTS = 10 #Number of events produced (1000)

PROD\_PID = 443013 # For PROD\\_MODE 1 through 4, this selects the channel to be produced, in PDG notation. Currently supported options are list below. (443013)

RND\_SEED = 34533 # Seed for random number generator. (34533)

BREAKUP\_MODE = 5 # Specifies the way nuclear break-up is handled. This option only works for lead or gold. It has no meaning in proton-proton or proton-nucleus collisions

1 = hard sphere nuclei (no hadronic break-up if impact parameter is greater than the sum of nuclear radii, no restriction on Coulomb break-up).

2 = requires Coulomb break-up of both nuclei, with no restriction on the number of neutrons emitted by either nucleus (XnXn).

3 = requires Coulomb break-up of both nuclei, but requires that a single neutron is emitted from each nucleus (1n1n).

4 = requires Coulomb break-up of neither nucleus. (0n0n)

5 = requires that there be no hadronic break up, no restriction on Coulomb break-up (This is similar to option 1, but with the actual hadronic interaction probability).

6 = requires Coulomb break up of one or both nuclei, with no restriction on the number of neutrons emitted (XnXn + 0nXn + Xn0n).

7 = requires Coulomb break up of only one nucleus, with no restriction on the number of neutrons emitted (0nXn+ Xn0n).

8 = selectable input parameter range (i.e. for peripheral collisions, not UPCs) regardless of nuclear breakup. Fixed input range between BMAX and BMIN (set by two otherwise optional cards, below)

INTERFERENCE = 0 # Specifies whether interference based on the ambiguity of which nucleus emits the photon is included. The effect of this interference is only visible at very small transverse momentum. 0 = interference off, 1 = interference on. (0)

IF\_STRENGTH = 1. # If interference is turned on, specifies the percentage of interference. The range is -1.0 – 1.0.; 1 is the standard value for ion-ion collisions, while -1.0 is expected for proton-antiproton collisions. (1)

INT\_PT\_MAX = 0.24 # Used only when the interference option above is turned on. This specifies the maximum transverse momentum considered, in GeV/c. (0.24)

INT\_PT\_N\_BINS = 120 # Used only when the interference option above is turned on. This specifies the number of bins in transverse momentum to use. (120)

INT\_PT\_WIDTH = 0 #Used only when the interference option above is turned on. This specifies the width of bins in transverse momentum to use. (0)

XSEC\_METHOD\* = 0 #Determines which method is used to calculate the cross-section for  cross-sections. XSEC\_METHOD=0 is faster, but works only for symmetric collisions (*i.e.* with identical nuclei). XSEC\_METHOD=1 always works, but is slower. (0)

BSLOPE\_DEFINITION\*=0 Used for proton and nucleon (i. e. incoherent nuclear) collisions to set the t-spectrum, dN/dt=exp(-bt). When BSLOPE\_DEFINITION=1, then the slope is determined by BSLOPE\_VALUE (below). When BSLOPE\_DEFINITION=2, the slope is calculated as a function of p center of mass energy per the H1 analysis, Eur. Phys. J. C46, 585 (2006):

b=4.63/GeV2 + 4ln(Wp/90 GeV)

The default value, BSLOPE\_DEFINITION=0 has no effect.

Note that this affects the t-slope only; it does not affect the total cross-section

BSLOPE\_VALUE\* WHEN BSLOPE\_DEFINITION=1, this determines the exponential slope for dN/dt=exp(-BSLOPE\_VALUE\*t)

SELECT\_IMPULSE\_VM When set =1, performs an impulse approximation calculation (this ignores most nuclear physics, including shadowing). Default=0; no change

QUANTUM\_GLAUBER When set =1, perform a quantum Glauber calculation, rather than classical, which is the default (or when set =0)

BMIN Needed for Breakup\_mode=8. Sets the minimum impact parameter

BMAX Needed for Breakup mode=8. Sets sthe maximum impact parameter.

OUTPUT\_HEADER Adds a header to the output file. This header will contain various input parameters. (1 for header, 0 for no header, default is no header)

The physics constants used by STARlight can be set with the following parameters:

deuteronSlopePar deuteron slope parameter (effective temperature)

[(GeV/c)^-2]

protonMass mass of the proton [GeV/c^2]

pionChargedMass mass of the pi^+/- [GeV/c^2]

pionNeutralMass mass of the pi^0 [GeV/c^2]

kaonChargedMass mass of the K^+/- [GeV/c^2]

mel mass of the e^+/- [GeV/c^2]

muonMass mass of the mu^+/- [GeV/c^2]

tauMass mass of the tau^+/- [GeV/c^2]

f0Mass mass of the f\_0(980) [GeV/c^2]

f0Width width of the f\_0(980) [GeV/c^2]

f0BrPiPi branching ratio f\_0(980) -> pi^+ pi^- and pi^0 pi^0

etaMass mass of the eta [GeV/c^2]

etaWidth width of the eta [GeV/c^2]

etaPrimeMass mass of the eta' [GeV/c^2]

etaPrimeWidth width of the eta' [GeV/c^2]

etaCMass mass of the eta\_c [GeV/c^2]

etaCWidth width of the eta\_c [GeV/c^2]

f2Mass mass of the f\_2(1270) [GeV/c^2]

f2Width width of the f\_2(1270) [GeV/c^2]

f2BrPiPi [GeV/c] f\_2(1270) -> pi^+ pi^-

a2Mass mass of the a\_2(1320) [GeV/c^2]

a2Width width of the a\_2(1320) [GeV/c^2]

f2PrimeMass mass of the f'\_2(1525) [GeV/c^2]

f2PrimeWidth width of the f'\_2(1525) [GeV/c^2]

f2PrimeBrKK branching ratio f'\_2(1525) -> K^+ K^- and K^0

K^0bar

zoverz03Mass mass of four-quark resonance (rho^0 pair

production) [GeV/c^2]

f0PartialggWidth partial width f\_0(980) -> g g [GeV/c^2]

etaPartialggWidth partial width eta -> g g [GeV/c^2]

etaPrimePartialggWidth partial width eta' -> g g [GeV/c^2]

etaCPartialggWidth partial width eta\_c -> g g [GeV/c^2]

f2PartialggWidth partial width f\_2(1270) -> g g [GeV/c^2]

a2PartialggWidth partial width a\_2(1320) -> g g [GeV/c^2]

f2PrimePartialggWidth partial width f'\_2(1525) -> g g [GeV/c^2]

zoverz03PartialggWidth partial width four-quark resonance -> g g (rho^0

pair production) [GeV/c^2]

f0Spin spin of the f\_0(980)

etaSpin spin of the eta

etaPrimeSpin spin of the eta'

etaCSpin spin of the eta\_c

f2Spin spin of the f\_2(1270)

a2Spin spin of the a\_2(1320)

f2PrimeSpin spin of the f'\_2(1525)

zoverz03Spin spin of the four-quark resonance -> g g (rho^0

pair production)

axionSpin spin of the axion

rho0Mass mass of the rho^0 [GeV/c^2]

rho0Width width of the rho^0 [GeV/c^2]

rho0BrPiPi branching ratio rho^0 -> pi^+ pi^-

rho0PrimeMass mass of the rho'^0 (4 pi^+/- final state)

[GeV/c^2]

rho0PrimeWidth width of the rho'^0 (4 pi^+/- final state)

[GeV/c^2]

rho0PrimeBrPiPi branching ratio rho'^0 -> pi^+ pi^-

OmegaMass mass of the omega [GeV/c^2]

OmegaWidth width of the omega [GeV/c^2]

OmegaBrPiPi branching ratio omega -> pi^+ pi^-

PhiMass mass of the phi [GeV/c^2]

PhiWidth width of the phi [GeV/c^2]

PhiBrKK branching ratio phi -> K^+ K^-

JpsiMass mass of the J/psi [GeV/c^2]

JpsiWidth width of the J/psi [GeV/c^2]

JpsiBree branching ratio J/psi -> e^+ e^-

JpsiBrmumu branching ratio J/psi -> mu^+ mu^-

JpsiBrppbar branching ratio J/psi -> p pbar

Psi2SMass mass of the psi(2S) [GeV/c^2]

Psi2SWidth width of the psi(2S) [GeV/c^2]

Psi2SBree branching ratio psi(2S) -> e^+ e^-

Psi2SBrmumu branching ratio psi(2S) -> mu^+ mu^-

Upsilon1SMass mass of the Upsilon(1S) [GeV/c^2]

Upsilon1SWidth width of the Upsilon(1S) [GeV/c^2]

Upsilon1SBree branching ratio Upsilon(1S) -> e^+ e^-

Upsilon1SBrmumu branching ratio Upsilon(1S) -> mu^+ mu^-

Upsilon2SMass mass of the Upsilon(2S) [GeV/c^2]

Upsilon2SWidth width of the Upsilon(2S) [GeV/c^2]

Upsilon2SBree branching ratio Upsilon(2S) -> e^+ e^-

Upsilon2SBrmumu branching ratio Upsilon(2S) -> mu^+ mu^-

Upsilon3SMass mass of the Upsilon(3S) [GeV/c^2]

Upsilon3SWidth width of the Upsilon(3S) [GeV/c^2]

Upsilon3SBree branching ratio Upsilon(3S) -> e^+ e^-

Upsilon3SBrmumu branching ratio Upsilon(3S) -> mu^+ mu^-

The following parameters are used only when interfacing with the PYTHIA and/or DPMJET interfaces:

MIN\_GAMMA\_ENERGY = 6 #Allows the user to set the minimum photon energy (in GeV) in the rest frame of the target nucleus. The default is 6.0 GeV and it should never be set below this value since DPMJET was not designed to handle low energy interactions.

MAX\_GAMMA\_ENERGY = 600000

#Allows the user to set the maximum photon energy (in GeV) in the rest frame of the target nucleus. The default is 60000.0 GeV.

PYTHIA\_PARAMS = ““ #Used to supply input parameters to the PYTHIA interface. This takes a string to pass on semi-colon separated parameters to PYTHIA 6. eg: "mstj(1)=0;paru(13)=0.1" (the default is a blank string " ")

PYTHIA\_FULL\_EVENT\_RECORD = 1

#Determines whether the full event record from PYTHIA is written to slight.out. true = yes, false = no (false). The additional information added is as follows: daughter production vertex (x [mm], y [mm], z [mm], t [mm/c]), mother1, mother2, daughter1, daughter2, PYTHIA particle status code. PYTHA 8 Particle Properties page describes in more detail the properties of mother, daughter, and status code designations.

-------------------------------------------------------------------------

Channels of Interest:

2-Photon Channels

Currently supported 2-photon (prod. mode = 1) channel options:

jetset id particle

---------------------------------

221 eta

331 eta-prime

441 eta-c

9010221 f0(975)

225 f2(1270)

115 a2(1320)

335 f2(1525)

33 rho0 pair

11 e+/e- pair

13 mu+/mu- pair

15 tau+/tau- pair

88 axion-like particle (ALP)

Process 88 refers to the single production of a hypothetical axion-like

particle (ALP), which decays to a pair of photons. The ALP mass has to be

specified by the user through the parameter AXION\_MASS. The narrow width

approximation is assumed here, with a fixed axion decay constant of

\Lambda=1 TeV. (See equation (1) of arXiv:1607.06083 for the appropriate

conventions.) The cross section can be then rescaled to arbitrary \Lambda, as long as the narrow width approximation remains valid.

Pomeron-Photon Channels

Currently supported vector meson (prod. mode = 2/3/4) options:

jetset id particle

---------------------------------

113 rho0

113011 rho0 --> e+e-

113013 rho0 --> mu+mu-

223 omega

223211111 omega --> pi+pi-pi0

333 phi

443011 J/psi --> e+e-

443013 J/Psi --> mu+mu-

4432212 J/psi --> proton antiproton

444011 Psi(2S) --> e+e-

444013 Psi(2S) --> mu+mu-

553011 Upsilon(1S) --> e+e-

553013 Upsilon(1S) --> mu+mu-

554011 Upsilon(2S) --> e+e-

554013 Upsilon(2S) --> mu+mu-

555011 Upsilon(3S) --> e+e-

555013 Upsilon(3S) --> mu+mu-

913 rho0 + direct pi+pi- (with interference). The direct pi+pi- fraction is from the ZEUS results, EPJ C2 p247 (1998)

999 four-prong final states (rho’-like to pi+pi-pi+pi-)

DPMJET:

Simulation of photonuclear interactions with STARlight is possible through an interface with DPMJet. These interfaces can be enabled through options passed to cmake during the configuration process. [Depreciated: Using Pythia 6 as a substitute for DPMJet]

The gfortran compiler is required to use the photonuclear interfaces.

=============== 1. Photonuclear interactions with DPMJet ===============

------- 1.1. Obtaining and installing DPMJet -------

The DPMJet package can be obtained by contacting the authors as explained here: <http://sroesler.web.cern.ch/sroesler/dpmjet3.html>

Once you have the code proceed with these steps:

Change the line containing the OPT variable in the DPMJet Makefile:

OPT = -c -C -std=legacy -O -O3 -g -fexpensive-optimizations -funroll-loops -fno-automatic -fbounds-check -v -fPIC

------------- 64-bit -------------

Make sure that all -m32 options are removed from the Makefile.

Unfortunately, the DPMJet package depends on a floating point exception trap implementation, and only a 32-bit version of that is included in the package, which needs to be replaced. An example implementation can be found here: <http://www.arsc.edu/arsc/support/news/hpcnews/hpcnews376/>

Under "Fortran Floating Point Traps for Linux" there is a code example. A file based on this, fpe.c, can be found in the external/ directory in STARlight. Move that to your DPMJet directory to replace the original file and run:

$ gcc -o fpe.o fpe.c

**Note**: if the above command returns the following error:

*/usr/lib/../lib64/crt1.o: In function `\_start':*

*(.text+0x20): undefined reference to `main'*

*/tmp/ccs2CQsd.o: In function `enable\_exceptions\_':*

*fpe.c:(.text+0xe): undefined reference to `feenableexcept'*

*collect2: error: ld returned 1 exit status*

**Try**: gcc fpe.c -Wall -g -c

feenableexcept is a gcc extension and gcc may need all of the headers present.

------------- End 64-bit -------------

Then in the DPMJet directory run:

$ make

Note: When compiling at RCAS(BNL), needed to change g77  gfortran, needed to install fluka and setenv FLUPRO /path/to/fluka, and modify phojet before compiling. The changes for phojet is at line 29875, from:

PRINT LO,'PHO\_DIFSLP:ERROR: this option is not installed !'

to:

WRITE(LO,'(/1X,A,I2)')

& 'PHO\_DIFSLP:ERROR: this option is not installed

& !',ISWMDL(13)

------------ 1.2. Compiling Starlight with DPMJet interface ------------

To enable the compilation of the DPMJet interface please follow these steps:

CMake uses an environment variable $DPMJETDIR to locate the DPMJet object files, so define it.

$ export DPMJETDIR=<path to dpmjet>

Then create a build directory for STARlight

$ mkdir <build-dir>

and change into it

$ cd <build-dir>

Run CMake with the option to enable DPMJet

$ cmake <path-to-starlight-source> -DENABLE\_DPMJET=ON

Then build it

$ make

Note: When compiling at RCAS(BNL), needed to add the gfortran library to the CMakeLists.txt and left it there.

----------- 1.3. Running Starlight with DPMJet interface -----------

To run Starlight with the DPMJet interface a couple of files are needed in the directory where you want to run Starlight.

The files needed are:

**slight.in** (Starlight config file. An example suitable for DPMJet can be found in config/slight.in.dpmjet)

**my.input** (DPMJet config file. An example can be found in config/my.input)

**dpmjet.dat** (Can be found in the DPMJet source directory)

In the slight.in file the relevant production modes (PROD\_MODE) for DPMJET is:

5: A+A single excitation

6: A+A double excitation

7: p+A single excitation

In addition the minimum and maximum gamma energies must be set. These must be within the interval set in the my.input file.

**To run:**

$ ./starlight < my.input

[DPMJET reads from direct input/interactive]

# Output

STARlight outputs an ASCII file named slight.out.

If OUTPUT\_HEADER = 1 (set in input file), then there will be a header at the beginning of the output file followed by a list of events. If OUTPUT\_HEADER = 0, or if OUTPUT\_HEADER is not set, then there will be no header in the output file and the file will start with the list of events.

If there is a header, it will be three lines, with the following format:

**CONFIG\_OPT:** prod\_mod particle\_id nevents q\_glauber impulse seed

where prod\_mod indicates if a wide or narrow resonance has been used, particle\_id specifies the vector meson species (and decay channel) being produced, nevents indicates the total number of events in the simulation, q\_glauber indicates if a quantum (=1) or classical (=0) Glauber has been selected, impulse indicates if the nuclear effects are being modelled (=0) or a simple impulse approx. is employed, and finally seed records the random number seed used when initializing the Monte Carlo. The config opt line is followed by two lines with brief descriptions of beams in the collision, with the format:

**BEAM\_1(2):** beam1(2)Z beam1(2)A beam1(2)LorentzGamma

where beam1(2)Z is the is the charge of the particles in beam 1(2), beam1(2)A indicates theatomic number of beam 1(2) and beam1(2)LorentzGamma is the Lorentz gamma factor associated to beam 1(2)

For each event, a summary line is printed, with the format

**EVENT**: n ntracks nvertices ,

where n is the event number (starting with 1), ntracks is the number of tracks in the event, and nvertices is the number of vertices in the event (STARlight does not currently produce events with more than one vertex).

EVENT line is followed by a description of the vertex, with the format

**VERTEX**: x y z t nv nproc nparent ndaughters ,

where x, y, z and t are the 4-vector components of the vertex location, nv is the vertex number, nproc is a number intended to represent physical process (always set to 0), nparent is the track number of parent track (0 for primary vertex) and ndaughters is the number of daughter tracks from this vertex.

This is followed by a series of lines describing each of the daughter tracks emanating from this vertex. Each track line has the format

**TRACK**: GPID px py py nev ntr stopv PDGPID ,

where GPID is the Geant particle id code, px, py and pz are the three vector components of the track's momentum, nev is the event number, ntr is the number of this track within the vertex (starting with 0), stopv is the vertex number where track ends (0 if track does not terminate within the event), and PDGPID is the Monte Carlo particle ID code endorsed by the Particle Data Group.

Class Diagram



File Descriptions

Readme.pdf

[This file.] provides information on the installation, operation, and construction of STARlight.

CMakeLists.txt

controls STARlight compilation. For details, please see above in [Installation](#Installation). This is the default/supported compilation method.

Makefile

A sample Makefile for compilation on \*nix systems. This file is not actively supported. Please use CMake.

starlightconfig.h.in

passes on some compiler settings; such as enabling the Pythia/DPMJet sections within the source code.

starlightDoxyfile.conf

Doxygen configuration file.

CMake Modules:

FindPythia8.cmake

used by CMake to find the Pythia 8 files needed to compile STARlight with Pythia 8 dependent options enabled. It searches for: Pythia.h, Index.xml, libpythia8

FindPythia6.cmake

used by CMake to find the Pythia 6 files needed to compile STARlight with Pythia 6 dependent options enabled. It searches for: libPythia6. *Pythia 6 functionality has been deprecated.*

FindDPMJet.cmake

used by CMake to find the DPMJET files needed to compile STARlight with DPMJET dependent options enabled. It searches for: dpmjet3.0-5.o, pythia6115dpm3v1.o, and phojet1.12-35c4.o

FindROOT.cmake

used by CMake to find the ROOT files needed to compile STARlight with ROOT dependent options enabled. It searches for: root-config. root-config is then used to set the rest of the paths/options needed to enable ROOT within STARlight.

CommonMacros.cmake

A collection of useful cmake macos.

FindLHAPDF.cmake

used by CMake to find the LHAPDF dependent options enabled. This was necessary for older versions of Pythia8, but this is no longer the case. However, this file is being kept in the distribution for users that would like to re-enable it. It searches for: Pythia.h and liblhapdfdummy

Config files:

my.input

A sample DPMJET configuration file.

slight.in

A sample STARlight input file, to select the desired final state and associated options. The section [Input](#Input) has more information.

slight.in.dpmjet

A sample slight.in file to use the DPMJET options (eg: PROD\_MODE = 5, 6, 7, and MIN\_GAMMA\_ENERGY, and MAX\_GAMMA\_ENERGY.).

slight.in.ee\_rhic

A sample slight.in file for e+e- production by Au-Au at top RHIC energies

slight.in.jpsi\_lhc

A sample slight.in file for J/ production by Pb-Pb at the LHC.

slight.in.pPb\_lhc

A sample slight.in file for J/ production by p-Pb at the LHC.

slight.in.rho\_rhic

A sample slight.in file for  production by Au-Au at top RHIC energies.

dpmjet:

dpmjetint.f

This is aDPMJET library, used in the CMakeLists.txt file to link when enabling DPMJET.

external:

fpe.c

corrects for the floating point trap differences between 32 and 64-bit. The [DPMJET section](#FloatingTrap) has more information.

pythia6:

pythiainterface.h

interfaces Pythia6 with STARlight. *Pythia 6 functionality has been deprecated.*

utils:

Ana.C

This macro runs Analyze.cxx, which takes as input an ASCII STARlight output file, slight.out, and creates a standard set of histograms, which are stored in histograms.root

Analyze.cxx

This macro reads in a starlight output file and creates histograms of the p\_T and rapidity of the daughters, as well as the p\_T, rapidity and mass of the parent. It assumes there are only 2 daughter tracks that are electrons, muons, or pions. The histograms for the daughter particles are called fPt2, fPt2, fRap1, and fRap2. Parent histograms are created for each possible daughter species (e.g., parent p\_T histograms are created with the names fPtEl, fPtMu, and fPtPi), but only the ones corresponding to the actual daughter particle are filled. The histograms are saved in a file called histograms.root.

To use this Analyze.cxx, modify the file Ana.C to call your input file (as downloaded, it calls slight.out) and the number of events you wish to process (as downloaded, it processes 20 events). Then open root and type ".x Ana.C" .

Analyze.h

The header file for Analyze.cxx and Ana.C.

AnalyzeTree.cxx

This macro reads the starlight.root file produced by ConvertStarlightAsciiToTree.C, which contains TLorentzVectors for the parents and a TClonesArray of TLorentzVectors for the daughters.It creates histograms of the p\_T and rapidity of the daughters, as well as the p\_T, rapidity and mass of the parent. While the parents may have been created as the vector sum of any number of daughter particles, this macro currently produces histograms for only the first two daughter particles. The daughter histograms are called D1Pt, D2Pt, D1Rapidity, and D1Rapidity. Parent histograms are named ParentPt, ParentRapidity, and ParentMass. The histograms are stored in starlight\_histos.root.

To use Analyzetree.cxx, first run ConvertStarlightAsciiToTree.C to produce the starlight.root file. If needed, modify the file AnalyzeTree.h to call your input file (as downloaded, it calls starlight.root). Then open root and type .x AnaTree.C .

AnalyzeTree.h

The header file for AnalyzeTree.cxx.

AnaTree.C

compiles and runs AnalyzeTree.cxx, which takes as input the starlight.root file produced by ConvertStarlightAsciiToTree.cxx output histograms are stored in starlight\_histos.root

ConvertStarlightAsciiToTree.C

reads a starlight output file (default name slight.out) and creates a root file with TLorentzVectors for the parent and a TClonesArray of TLorentzVectors for the daughter particles. The output is stored in a root file (default name starlight.root) with one branch labeled "parent" and the other labeled "daughters". Any number of daughter tracks can be accommodated. Daughter species currently accommodated are: electrons, muons, charged or neutral pions, charged or neutral kaons, and protons.

To use AnaTree.C, open root and then type .x ConvertStarlightAsciiToTree.C("inputfilename", "outputfilename") The root file produced can be examined in a root TBrowser.

A macro to read this root file and make some standard plots is also provided. This macro is called AnalyzeTree.cxx; it can be compiled and run with the AnaTree.C macro by opening root and typing .x AnaTree.C()

Source Files:

beam.cpp

generates the beam class, which inherits from the nucleus class (cf. [nucleus.cpp](#nucleus_cpp)). The object represents an accelerated nucleus, or a beam.

**Functions**:

beam::beam

beam::~beam

beam::photonFlux *// calculates the “photon density” given the impact parameter and energy.*

beambeamsystem.cpp

represents the colliding system of interest.

**Functions**:

beamBeamSystem::beamBeamSystem

beamBeamSystem::~beamBeamSystem

beamBeamSystem::probabilityOfBreakup

beamBeamSystem::generateBreakupProbabilities

beamBeamSystem::probabilityOfHadronBreakup

beamBeamSystem::probabilityOfPhotonBreakup

bessel.cpp

calculate modified Bessel functions of the first and second kind.

**Functions:**

bessel::besI0

bessel::dbesk0

bessel::dbesk1

bessel::besI1

eventchannel.cpp

inherits from readLuminosity. It is a base for class for functions to produce events that is overloaded by other classes (Gammagammaleptonpair, Gammagammasingle, Gammaavectormeson, starlightDpmJet, and starlightPythia).

**Functions:**

eventChannel::eventChannel

eventChannel::~eventChannel

eventChannel::transform // Lorentz Tranforms the frame

eventChannel::pseudoRapidity // calculates the pseudorapidity with the input from px, py, and pz

eventfilewriter.cpp

writes event information in the output file.

**Functions:**

eventFileWriter::eventFileWriter

eventFileWriter::~eventFileWriter

eventFileWriter::writeEvent

filewriter.cpp

The base class for eventFileWriter, which is writes event information in the output file.

**Functions:**

fileWriter::fileWriter()

fileWriter::~fileWriter()

fileWriter::open

fileWriter::open(filename)

fileWriter::close

gammaaluminosity.cpp

contains the photonNucleusLuminosity class, which inherits from photonNucleusCrossSection. It calculates the differential cross-section for gamma-A interactions.

**Functions:**

photonNucleusLuminosity::photonNucleusLuminosity

photonNucleusLuminosity::~photonNucleusLuminosity

photonNucleusLuminosity::photonNucleusDifferentialLuminosity //Calculates and outputs the differential luminosity

photonNucleusLuminosity::pttablegen // Calculates the pt spectra for VM production with interference per S. Klein and J. Nystrand, Phys. Rev Lett. 84, 2330 (2000).

photonNucleusLuminosity::vmsigmapt //calculates th effect of the nuclear form factor on the pt spectrum, for use in interference calculations. It calculates the cross section suppression SIGMAPT(PT) as a function of pt. The input pt values come from pttable.inc

photonNucleusLuminosity::nofe //calculates the ‘photon density’d^2N\_gamma/db^2

gammaavm.cpp

isresponsible for classes Gammaavectormesion, Gammaanarrowvm, and Gammaawidevm. Both Gammaanarrowvm and Gammaawidevm inherit from Gammaavectormeson, which inherits from eventChannel. The classes are responsible for generating and decaying the vector mesons produced by photon-nucleus interactions.

**Functions:**

Gammaavectormeson::Gammaavectormeson

Gammaavectormeson::~Gammaavectormeson

Gammaavectormeson::pickwy //responsible for selecting the events center of mass energy and rapidity

Gammaavectormeson::twoBodyDecay // This routine decays a particle into two particles of mass mdec, taking spin into account

Gammaavectormeson::fourBodyDecay // decays a particle into four particles with isotropic angular distribution

Gammaavectormeson::getDaughterMass //returns the daughter particles mass, & the final particles id...

Gammaavectormeson::getTheta //This depends on the decay angular distribution

Gammaavectormeson::getWidth

Gammaavectormeson::getMass

Gammaavectormeson::getSpin //it’s a VM, returns 1

Gammaavectormeson::momenta // calculates momentum and energy of vector meson given W and Y, without interference.

Gammaavectormeson::pTgamma //finds the photon pT

Gammaavectormeson::vmpt // calculates momentum and energy of a vector meson given W and Y, including interference. It gets the pt distribution from a lookup table.

produceEvent

pseudorapidity

Gammaanarrowvm::Gammaanarrowvm

Gammaanarrowvm::~Gammaanarrowvm

Gammaanarrowvm::gammaaincoherentvm

Gammaawidevm::Gammaawidevm

Gammaawidevm::~Gammaawidevm

gammagammaleptonpair.cpp

inherits from eventChannel. It calculates the lepton pair’s cross-section and generates and decayes the lepton pairs.

**Functions:**

Gammagammaleptonpair::Gammagammaleptonpair

Gammagammaleptonpair::~Gammagammaleptonpair

Gammagammaleptonpair::twoLeptonCrossSection // calculates section for 2-particle decay, per, see STAR Note 243, Eq. 9. It calculates the 2-lepton differential cross section

Gammagammaleptonpair::twoMuonCrossSection // gives the two muon cross section as a function of Y&W, per G.Soff et. al Nuclear Equation of State, part B, 579

Gammagammaleptonpair::pickw // Picks a w for the 2- photon calculation.

Gammagammaleptonpair::picky // Picks a y given a W

Gammagammaleptonpair::pairMomentum // calculates px,py,pz,and E given w and y

Gammagammaleptonpair::pp\_1 // For beam 1, returns a random momentum drawn from from pp\_1(E) distribution

Gammagammaleptonpair::pp\_2 // For beam 2, returns a random momentum drawn from from pp\_2(E) distribution

Gammagammaleptonpair::twoBodyDecay //decays a particle into two particles of mass mdec, taking spin into account

Gammagammaleptonpair::thetalep // calculates the cross-section as a function of angle for a given W and Y, for the production of two muons or taus, per Brodsky et al. PRD 1971, 1532 equation 5.7

Gammagammaleptonpair::produceEvent //returns the vector with the decay particles inside

Gammagammaleptonpair::calculateTable //calculates the tables that are used elsewhere in the Monte Carlo the tau decay follows V-A theory, 1 - 1/3 cos(theta)the energy of each of the two leptons in tau decay is calculated using formula 10.35 in “Introduction to elementary particles by D. Griffiths,” which assumes that the mass of the electron is 0. The maximum electron energy in in such a system is 0.5 \* mass of the tau

Gammagammaleptonpair::tauDecay // assumes that the tauons decay to electrons and calculates the directons of the decays

Gammagammaleptonpair::getMass

Gammagammaleptonpair::getWidth

Gammagammaleptonpair::getSpin

gammagammasingle.cpp

inherits from eventChannel. It calculates the cross-section for single mesons and generates and decays the single mesons from gamma-gamma interactions. It also generates single mesons which are then decayed by Pythia 8.

**Functions:**

Gammagammasingle::Gammagammasingle

Gammagammasingle::~Gammagammasingle

Gammagammasingle::singleCrossSection // calculates the cross-section in the narrow-width approximation, per STAR Note 243, Eq. 8

Gammagammasingle::pickw // picks a w for the 2-photon calculation.

Gammagammasingle::picky

Gammagammasingle::parentMomentum // calculates px,py,pz,and E given w and y

Gammagammasingle::pp\_1 // For beam 1, returns a random momentum drawn from from pp(E) distribution

Gammagammasingle::pp\_2 // For beam 2, returns a random momentum drawn from from pp(E) distribution

Gammagammasingle::twoBodyDecay //decays a particle into two particles of mass mdec, taking spin into account

Gammagammasingle::produceEvent

Gammagammasingle::getMass

Gammagammasingle::getSpin

incoherentPhotonNucleusLuminosity.cpp

is responsible for the incoherentPhotonNucleusLuminosity class and inherits from photonNucleusCrossSection. It houses the differential luminosity calculation for incoherent gamma-A interactions.

**Functions:**

incoherentPhotonNucleusLuminosity::incoherentPhotonNucleusLuminosity

incoherentPhotonNucleusLuminosity::~incoherentPhotonNucleusLuminosity

incoherentPhotonNucleusLuminosity::incoherentPhotonNucleusDifferentialLuminosity

incoherentPhotonNucleusLuminosity::nofe //Function for the calculation of the "photon density".

incoherentVMCrossSection.cpp

inherits from photonNucleusCrossSection. It calculates the cross-section for incoherent photon-nucleus interactions.

**Functions:**

incoherentVMCrossSection::incoherentVMCrossSection

incoherentVMCrossSection::~incoherentVMCrossSection

incoherentVMCrossSection::crossSectionCalculation // calculates the vector meson cross section assuming a narrow resonance. For reference, see STAR Note 386.

inputParameters.cpp

sets and stores STARlight’s input parameters.

**Functions:**

inputParameters::inputParameters

inputParameters::~inputParameters

inputParameters::init

inputParameters::configureFromFile

inputParameters::print

inputParameters::write

inputParameters::parameterValueKey

inputParser.cpp

parses the input files and stores the information in the inputParameters.

**Functions:**

inputParser::inputParser()

inputParser::~inputParser()

inputParser::parseFile

inputParser::parseString

inputParser::addIntParameter

inputParser::addUintParameter

inputParser::addFloatParameter

inputParser::addDoubleParameter

inputParser::addBoolParameter

inputParser::addStringParameter

inputParser::printParameterInfo

inputParser::validateParameters

lorentzvector.cpp

holds Lorentz 4-vectors.

**Functions:**

lorentzVector::lorentzVector

lorentzVector::~lorentzVector

SetXYZT

main.cpp

the “main” file/function—where the program starts.

narrowResonanceCrossSection.cpp

inherits from photonNucleusCrossSection. It calculates the cross-section for narrow resonance vector mesons.

**Functions:**

narrowResonanceCrossSection::narrowResonanceCrossSection

narrowResonanceCrossSection::~narrowResonanceCrossSection

narrowResonanceCrossSection::crossSectionCalculation // calculates the vector meson cross section assuming a narrow resonance, per STAR Note 386.

nBodyPhaseSpaceGen.cpp

is responsible for the kinematics used in the four-prong decays.

**Functions:**

nBodyPhaseSpaceGen::nBodyPhaseSpaceGen

nBodyPhaseSpaceGen::~nBodyPhaseSpaceGen

nBodyPhaseSpaceGen::setDecay // sets decay constants and prepares internal variables

nBodyPhaseSpaceGen::generateDecay// generates event with certain n-body mass and momentum and returns event weight general purpose function

nBodyPhaseSpaceGen::generateDecayAccepted// generates full event with certain n-body mass and momentum only, when event is accepted (return value = true) this function is more efficient, if only weighted evens are needed

nBodyPhaseSpaceGen::pickMasses// randomly choses the (n - 2) effective masses of the respective (i + 1)-body systems

nBodyPhaseSpaceGen::calcWeight// computes event weight (= integrand value) and breakup momenta uses vector of intermediate two-body masses prepared by pickMasses()

nBodyPhaseSpaceGen::calcEventKinematics// calculates complete event from the effective masses of the (i + 1)-body systems, the Lorentz vector of the decaying system, and the decay angles uses the break-up momenta calculated by calcWeight()

nBodyPhaseSpaceGen::estimateMaxWeight// calculates maximum weight for given n-body mass

nBodyPhaseSpaceGen::print

nucleus.cpp

defines the basis properties of a nucleus such as radius, form factor, and thickness.

**Functions:**

nucleus::nucleus

nucleus::~nucleus

nucleus::init

nucleus::nuclearRadius

nucleus::formFactor

nucleus::dipoleFormFactor

nucleus::thickness// calculates the nuclear thickness function per Eq. 4 in Klein and Nystrand, PRC 60

photonNucleusCrossSection.cpp

calculates the cross-section for coherent photon-Nucleus interactions.

**Functions:**

photonNucleusCrossSection::photonNucleusCrossSection

photonNucleusCrossSection::~photonNucleusCrossSection

photonNucleusCrossSection::getcsgA // returns the cross-section for photon-nucleus interaction producing vector mesons

photonNucleusCrossSection::photonFlux // gives the photon flux as a function of energy Egamma for arbitrary nuclei and gamma. The first time it is called, it calculates a lookup table which is used on subsequent calls. It returns dN\_gamma/dE (dimensions 1/E), not dI/dE energies are in GeV, in the lab frame

photonNucleusCrossSection::nepoint// gives the spectrum of virtual photons, dn/dEgamma, for a point charge q=Ze sweeping past the origin with velocity gamma, integrated over impact parameter from bmin to infinity, per Eq. 15.54 of Jacksons Classical Electrodynamics

photonNucleusCrossSection::sigmagp// gives the gamma-proton --> VectorMeson cross section. Wgp is the gamma-proton CM energy. Unit for cross section: fm\*\*2

photonNucleusCrossSection::sigma\_A// Nuclear Cross Section sig\_N,sigma\_A in (fm\*\*2)

photonNucleusCrossSection::sigma\_N// Nucleon Cross Section in (fm\*\*2)

photonNucleusCrossSection::breitWigner// uses simple fixed-width s-wave Breit-Wigner without coherent backgorund for rho’ (PDG '08 eq. 38.56)

pythiadecayer.cpp

links Pythia 8 and STARlight, and initalizes Pythia 8.

**Functions:**

pythiaDecayer::pythiaDecayer

pythiaDecayer::~pythiaDecayer

pythiaDecayer::init

pythiaDecayer::addParticle

pythiaDecayer::execute

randomgenerator.cpp

STARlight’s random number generator, using the same algorithm as ROOTs TRANDOM3 class. It is based on M. Matsumoto and T. Nishimura, Mersenne Twistor: A 623-dimensionally equidistributed uniform pseudorandom number generator. For more information see http://www.math.keio.ac.jp/~matumoto/emt.html

**Functions:**

randomGenerator::SetSeed

randomGenerator::Rndom

readinluminosity.cpp

reads in the luminosity tables from slight.txt, which is generated in the early stages of the program.

**Functions:**

readLuminosity::readLuminosity

readLuminosity::~readLuminosity

readLuminosity::read

spectrum.cpp

sets up functions needed to make cross-section calculations for general photonuclear interactions modeled with DPMJET.

**Functions:**

spectrum::spectrum

spectrum::generateKsingle

spectrum::generateKdouble

spectrum::drawKsingle

spectrum::drawKdouble

spectrum::generateBreakupProbabilities

spectrum::getFnSingle

spectrum::getFnDouble

spectrum::getTransformedNofe

sprectrumprotonnucleus.cpp

sets up functions needed to make cross-section calculations for general photonuclear interactions modeled with DPMJET.

**Functions:**

spectrumProtonNucleus::spectrumProtonNucleus

spectrumProtonNucleus::generateBreakupProbabilities

spectrumProtonNucleus::getSigma

starlight.cpp

initializes and then produces and decays events.

**Functions:**

starlight::starlight

starlight::~starlight

starlight::init

starlight::produceEvent

starlight::luminosityTableIsValid

starlight::createEventChannel

starlightdpmjet.cpp

hosts the class starlightDpmJet which inherits from the eventChannel class. It includes methods to generate diffractive events with DPMJET.

**Functions:**

starlightDpmJet::starlightDpmJet

starlightDpmJet::init

starlightDpmJet::produceEvent

starlightDpmJet::produceSingleEvent

starlightDpmJet::produceDoubleEvent

starlightparticle.cpp

is a container to store particle information.

**Functions:**

starlightParticle::starlightParticle

starlightParticle::~starlightParticle

starlightparticlecodes.cpp

**c**onverts jetset particle numbers to the corresponding GEANT code.

**Functions:**

starlightParticleCodes::jetsetToGeant

starlightpythia.cpp

inherits from the eventChannel class. It includes methods to calculate diffractive events with Pythia6. *Pythia 6 functionality has been deprecated.*

**Functions:**

starlightPythia::starlightPythia

starlightPythia::~starlightPythia

starlightPythia::init

starlightPythia::produceEvent

starlightStandalone.cpp

is used by Main.cpp and in turn calls methods from the starlight class.

**Functions:**

starlightStandalone::starlightStandalone

starlightStandalone::~starlightStandalone

starlightStandalone::init

starlightStandalone::run

starlightStandalone::boostEvent

twophotonluminosity.cpp

inherits from beamBeamSystem, and is responsible for calculating the two photon luminosity table based on W and Y.

**Functions:**

twoPhotonLuminosity::twoPhotonLuminosity

twoPhotonLuminosity::~twoPhotonLuminosity

twoPhotonDifferentialLuminosity

twoPhotonLuminosity::D2LDMDY

twoPhotonLuminosity::D2LDMDY\_Threaded

twoPhotonLuminosity::integral

twoPhotonLuminosity::radmul

twoPhotonLuminosity::integrand

twoPhotonLuminosity::Nphoton

upcevent.cpp

stores the final event information.

**Functions:**

upcEvent::upcEvent

upcEvent::operator=

upcEvent::operator+

upcEvent::boost

vector3.cpp

is a container for 3D-vectors.

**Functions:**

vector3::vector3

vector3::~vector3

vector3::SetVector

wideResonanceCrossSection.cpp

inherits from photnNucleusCrossSection. It is responsible for calculating the cross-section of vector mesons with a wide resonance (eg. Rho).

**Functions:**

wideResonanceCrossSection::wideResonanceCrossSection

wideResonanceCrossSection::~wideResonanceCrossSection

wideResonanceCrossSection::crossSectionCalculation // calculates the cross-section assuming a wide(Breit-Wigner) resonance.

Include Files:

beam.h //This class includes a single beam of nucleons

**Included in files**

[beambeamsystem.h](#beambeamsystem_h)

[twophotonluminosity.h](#twophotonluminosity_h)

[beam.cpp](#beam_cpp)

[gammaaluminosity.cpp](#gammaaluminosity_cpp)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[spectrumprotonnucleus.cpp](#sprectrumprotonnucleus_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

beam

~beam

rapidity

photonFlux

setBeamLorentzGamma

beambeamsystem.h //This class covers a coliding beam system

**Included in files**

[eventchannel.h](#eventchannel_h)

[gammaaluminosity.h](#gammaaluminosity_h)

[gammaavm.h](#gammaavm_h)

[gammagammasingle.h](#gammagammasingle_h)

[incoherentPhotonNucleusLuminosity.h](#incoherentPhotonNucleusLuminosity_h)

[photonNucleusCrossSection.h](#photonNucleusCrossSection_h)

[starlightpythia.h](#starlightpythia_h)

[twophotonluminosity.h](#twophotonluminosity_h)

[beambeamsystem.cpp](#beambeamsystem_cpp)

[gammaaluminosity.cpp](#gammaaluminosity_h)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[spectrum.cpp](#spectrum_cpp)

[spectrumprotonnucleus.cpp](#sprectrumprotonnucleus_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

beamBeamSystem

~beamBeamSystem

cmsBoost

beamLorentzGamma

beam1

beam2

probabilityOfBreakup

init

generateBreakupProbabilities

probabilityOfHadronBreakup

probabilityOfPhotonBreakup

bessel.h

**Included in files**

[beam.cpp](#beam_cpp)

[beambeamsystem.cpp](#beambeamsystem_cpp)

[bessel.cpp](#bessel_cpp)

[gammaaluminosity.cpp](#gammaaluminosity_cpp)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[photonNucleusCrossSection.cpp](#photonNucleusCrossSection_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

besI0

dbesk0

dbesk1

besI1

eventchannel.h

**Included in files**

[gammaavm.h](#gammaavm_h)

[gammagammaleptonpair.h](#gammagammaleptonpair_h)

[gammagammasingle.h](#gammagammasingle_h)

[starlight.h](#starlight_h)

[starlightdpmjet.h](#starlightdpmjet_h)

[starlightpythia.h](#starlightpythia_h)

[eventchannel.cpp](#eventchannel_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

eventChannel

~eventChannel

nmbAttempts ///< returns number of attempted events

nmbAccepted ///< returns number of accepted events

produceEvent

transform ///< Lorentz-transforms given 4-vector

pseudoRapidity ///< calculates pseudorapidity for given 3-momentum

eventfilewriter.h

**Included in files**

[eventfilewriter.cpp](#eventfilewriter_cpp)

[main.cpp](#main_cpp)

[starlight.cpp](#starlight_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

**Functions**

eventFileWriter

writeEvent /\*\* Write an UPC event to file \*/

writeFullPythiaInfo /\*\* Set if we want to write full pythia information \*/

filewriter.h

**Included in files**

[eventfilewriter.h](#eventfilewriter_h)

[eventfilewriter.cpp](#eventfilewriter_cpp)

[filewriter.cpp](#filewriter_cpp)

[main.cpp](#main_cpp)

[starlight.cpp](#starlight_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

**Functions**

fileWriter

~fileWriter

open //opens the file

setFileName//set the filename we’re writing to

gammaaluminosity.h

**Included in files**

[gammaaluminosity.cpp](#gammaaluminosity_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

photonNucleusLuminosity

~photonNucleusLuminosity

photonNucleusDifferentialLuminosity

vmsigmapt

nofe

pttablegen

gammaavm.h

**Included in files**

[gammaavm.cpp](#gammaavm_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

Gammaavectormeson

~Gammaavectormeson

produceEvent

pickwy

momenta

pTgamma

vmpt

twoBodyDecay

fourBodyDecay

getMass

getWidth

getTheta

getSpin

getDaughterMass

pseudoRapidity

Gammaanarrowvm

~Gammaanarrowvm

Gammaawidevm

~Gammaawidevm

Gammaaincoherentvm

~Gammaaincoherentvm

gammagammaleptonpair.h

**Included in files**

[gammagammaleptonpair.cpp](#gammagammaleptonpair_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

Gammagammaleptonpair

~Gammagammaleptonpair

twoLeptonCrossSection

calculateTable

produceEvent

twoMuonCrossSection

pickw

picky

pairMomentum

pp\_1

pp\_2

twoBodyDecay

thetalep

tauDecay

getMass

getWidth

getSpin

gammagammasingle.h

**Included in files**

[gammagammasingle.cpp](#gammagammasingle_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

Gammagammasingle

~Gammagammasingle

singleCrossSection

produceEvent

pickw

picky

parentMomentum

pp

twoBodyDecay

thephi

getMass

getWidth

getSpin

incoherentPhotonNucleusLuminosity.h

**Included in files**

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[starlight.cpp](#starlight_cpp)

**Functions**

incoherentPhotonNucleusLuminosity

~incoherentPhotonNucleusLuminosity

incoherentPhotonNucleusDifferentialLuminosity

nofe

incoherentVMCrossSection.h

**Included in files**

[gammaavm.cpp](#gammaavm_cpp)

[incoherentVMCrossSection.cpp](#incoherentVMCrossSection_cpp)

**Functions**

incoherentVMCrossSection

~incoherentVMCrossSection

crossSectionCalculation

inputParameters.h

**Included in files**

[beam.h](#beam_h)

[gammaaluminosity.h](#gammaaluminosity_h)

[incoherentPhotonNucleusLuminosity.h](#incoherentPhotonNucleusLuminosity_h)

[readinluminosity.h](#readinluminosity_h)

[starlightpythia.h](#starlightpythia_h)

[beam.cpp](#beam_cpp)

[beambeamsystem.cpp](#beambeamsystem_cpp)

[gammaaluminosity.cpp](#gammaaluminosity_cpp)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[inputParameters.cpp](#inputParameters_cpp)

[nucleus.cpp](#nucleus_cpp)

[readinluminosity.cpp](#readinluminosity_cpp)

[starlight.cpp](#starlight_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

parameterlist

add

validationKey

parameterbase

toString

operator<<

parameter

operator=

ptr

value

name

required

setValue

setName

setRequired

inputParameters

~inputParameters

init

configureFromFile

baseFileName

beam1Z

beam1A

beam2Z

beam2A

beamLorentzGamma

beam1LorentzGamma

beam2LorentzGamma

maxW

minW

nmbWBins

maxRapidity

nmbRapidityBins

ptCutEnabled

ptCutMin

ptCutMax

etaCutEnabled

etaCutMin

etaCutMax

productionMode

nmbEvents

prodParticleId

randomSeed

beamBreakupMode

interferenceEnabled

interferenceStrength

maxPtInterference

nmbPtBinsInterference

ptBinWidthInterference

coherentProduction

incoherentFactor

minGammaEnergy

maxGammaEnergy

pythiaParams

pythiaFullEventRecord

xsecCalcMethod

prodParticleType

prodParticleDecayType

interactionType

protonEnergy

setBaseFileName

setBeam1Z

setBeam1A

setBeam2Z

setBeam2A

setBeamLorentzGamma

setBeam1LorentzGamma

setBeam2LorentzGamma

setMaxW

setMinW

setNmbWBins

setMaxRapidity

setNmbRapidityBins

setPtCutEnabled

setPtCutMin

setPtCutMax

setEtaCutEnabled

setEtaCutMin

setEtaCutMax

setProductionMode

setNmbEvents

setProdParticleId

setRandomSeed

setBeamBreakupMode

setInterferenceEnabled

setInterferenceStrength

setMaxPtInterference

setNmbPtBinsInterference

setPtBinWidthInterference

setCoherentProduction

setIncoherentFactor

setMinGammaEnergy

setMaxGammaEnergy

setPythiaParams

setPythiaFullEventRecord

setXsecCalcMethod

setProdParticleType

setProdParticleDecayType

setInteractionType

setProtonEnergy

setParameter

print

write

parameterValueKey

instance

inputParser.h

**Included in files**

[inputParameters.h](#inputParameters_h)

[inputParameters.cpp](#inputParameters_cpp)

[inputParser.cpp](#inputParser_cpp)

**Functions**

inputParser

inputParser

parseFile/\*\* Parse a file \*/

parseString

addIntParameter

addUintParameter

addFloatParameter

addDoubleParameter

addBoolParameter

addStringParameter

printParameterInfo

validateParameters

\_parameter

operator==

operator<

printParameterInfo

addParameter

lorentzvector.h

**Included in files**

[nBodyPhaseSpaceGen.h](#nBodyPhaseSpaceGen_h)

[starlightparticle.h](#starlightparticle_h)

[lorentzvector.cpp](#lorentzvector_cpp)

**Functions**

lorentzVector

~lorentzVector

SetXYZT

SetPxPyPzE

GetPx

GetPy

GetPz

GetE

operator +=

operator -=

M2

M

BoostVector

Boost

operator <<

narrowResonanceCrossSection.h

**Included in files**

[narrowResonanceCrossSection.cpp](#narrowResonanceCrossSection_cpp)

[gammaavm.cpp](#gammaavm_cpp)

**Functions**

narrowResonanceCrossSection

~narrowResonanceCrossSection

crossSectionCalculation

nBodyPhaseSpaceGen.h

**Included in files**

[gammaavm.h](#gammaavm_h)

[nBodyPhaseSpaceGen.cpp](#nBodyPhaseSpaceGen_cpp)

**Functions**

Factorial

breakupMomentum

nBodyPhaseSpaceGen

~nBodyPhaseSpaceGen

setDecay

random

generateDecay

generateDecayAccepted

setMaxWeight

maxWeight

normalization

eventWeight

maxWeightObserved

resetMaxWeightObserved

estimateMaxWeight

eventAccepted

daughter

daughters

nmbOfDaughters

daughterMass

intermediateMass

breakupMom

cosTheta

phi

print

operator <<

pickMasses

calcWeight

pickAngles

calcEventKinematics

eventAccepted

nucleus.h

**Included in files**

[beam.h](#beam_h)

[beambeamsystem.h](#beambeamsystem_h)

[twophotonluminosity.h](#twophotonluminosity_h)

[gammaaluminosity.h](#gammaaluminosity_h)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[nucleus.cpp](#nucleus_cpp)

[spectrumprotonnucleus.cpp](#sprectrumprotonnucleus_cpp)

[starlightdpmjet.cpp](#starlightdpmjet_cpp)

[starlightpythia.cpp](#starlightpythia_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

nucleus

~nucleus

init

Z

A

nuclearRadius

formFactor

dipoleFormFactor

thickness

Q0

rho0

woodSaxonSkinDepth

fritiofR0

rws

photonNucleusCrossSection.h

**Included in files**

[gammaaluminosity.h](#gammaaluminosity_h)

[incoherentPhotonNucleusLuminosity.h](#incoherentPhotonNucleusLuminosity_h)

[incoherentVMCrossSection.h](#incoherentVMCrossSection_h)

[narrowResonanceCrossSection.h](#narrowResonanceCrossSection_h)

[wideResonanceCrossSection.h](#wideResonanceCrossSection_h)

[gammaavm.cpp](#gammaavm_cpp)

[photonNucleusCrossSection.cpp](#photonNucleusCrossSection_cpp)

**Functions**

photonNucleusCrossSection

~photonNucleusCrossSection

slopeParameter///< returns slope of t-distribution [(GeV/c)^{-2}]

getChannelMass ///< returns mass of the produced system [GeV/c^2]

getBNORM

luminosity//< returns luminosity [10^{26} cm^{-2} sec^{-1}]

getbbs///< returns beamBeamSystem

vmPhotonCoupling ///< vectormeson-photon coupling constant f\_v / 4 pi (cf. Eq. 10 in KN PRC 60 (1999) 014903)

getDefaultC

maxPhotonEnergy///< returns max photon energy in lab frame [GeV] (for vectormesons only)

crossSectionCalculation

getcsgA

photonFlux

sigmagp

sigma\_A

sigma\_N

breitWigner

nepoint

pythiadecayer.h

**Included in files**

[gammagammasingle.h](#gammagammasingle_h)

[pythiadecayer.cpp](#pythiadecayer_cpp)

**Functions**

pythiaDecayer

~pythiaDecayer

init// Initialize

addParticle// Add particle to current event

execute// Execute event and return starlight type event

pythiaDecayer

operator==

PythiaStarlight.h

**Included in files**

[starlight.cpp](#starlight_cpp)

**Functions**

pythiaStarlight

init

getPythia

randomgenerator.h

**Included in files**

[eventchannel.h](#eventchannel_h)

[gammaavm.h](#gammaavm_h)

[gammagammasingle.h](#gammagammasingle_h)

[nBodyPhaseSpaceGen.h](#nBodyPhaseSpaceGen_h)

[inputParameters.cpp](#inputParameters_cpp)

[randomgenerator.cpp](#randomgenerator_cpp)

[spectrum.cpp](#spectrum_cpp)

**Functions**

SetSeed

Rndom

randomGenerator

instance

readinluminosity.h

**Included in files**

[eventchannel.h](#eventchannel_h)

[gammaavm.h](#gammaavm_h)

[gammagammaleptonpair.h](#gammagammaleptonpair_h)

[gammagammasingle.h](#gammagammasingle_h)

[readinluminosity.cpp](#readinluminosity_cpp)

**Functions**

readLuminosity

~readLuminosity

read

reportingUtils.h

**Included in files**

[inputParser.h](#inputParser_h)

[nBodyPhaseSpaceGen.h](#nBodyPhaseSpaceGen_h)

[beam.cpp](#beam_cpp)

[beambeamsystem.cpp](#beambeamsystem_cpp)

[inputParameters.cpp](#inputParameters_cpp)

[main.cpp](#main_cpp)

[nucleus.cpp](#nucleus_cpp)

[photonNucleusCrossSection.cpp](#photonNucleusCrossSection_cpp)

[pythiadecayer.cpp](#pythiadecayer_cpp)

[starlight.cpp](#starlight_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

**Functions**

getClassMethod\_\_

printErr

printWarn

printInfo

svnVersion

printSvnVersion

compileDir

printCompilerInfo

operator <<

progressIndicator

trueFalse

yesNo

onOff

enDisabled

spectrum.h

**Included in files**

[spectrumprotonnucleus.h](#sprectrumprotonnucleus_h)

[starlightdpmjet.h](#starlightdpmjet_h)

[spectrum.cpp](#spectrum_cpp)

[starlightdpmjet.cpp](#starlightdpmjet_cpp)

**Functions**

spectrum // Spectrum must be constructed with beam-beam system, default constructor disallowed

generateKsingle // Generate a table of photon energy probabilities. Use NK+1 logarithmic steps between Et\_min and Eg\_max

generateKdouble // Generate a 2-D table of photon energy probabilities. Use NK+1 x NK+1 logarithmic steps between Et\_min and Eg\_max

drawKsingle // Get the energy of a single gamma @return energy of the gamma

drawKdouble // Get the energy of a single gamma @param egamma1 variable passed by reference to get the energy of the frst gamma @param egamma2 variable passed by reference to get the energy of the second gamma @return energy of the gamma

setBeamBeamSystem // Set the beam beam system

setMinGammaEnergy //Set the minimum gamma energy

setMaxGammaEnergy / Set the maximum gamma energy

setBmin //Set minimum impact parameter

setBMax //Set maximum impact parameter

generateBreakupProbabilities //Generate the hadron breakup probability table

getSigma ---1.05?

getTransformedNofe

getFnSingle

getFnDouble

sprectrumprotonnucleus.h

**Included in files**

spectrumprotonnucleus.cpp

starlightdpmjet.cpp

starlightpythia.cpp

**Functions**

spectrumProtonNucleus

getNucleonNucleonSigma --- 7.35?

generateBreakupProbabilities

getSigma

starlight.h

**Included in files**

[main.cpp](#main_cpp)

[starlight.cpp](#starlight_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

**Functions**

starlight

~starlight

init

produceEvent

configFileName

nmbAttempts

nmbAccepted

luminosityTableIsValid

createEventChannel

starlightconstants.h

**Included in files**

[eventchannel.h](#eventchannel_h)

[gammaavm.h](#gammaavm_h)

[gammagammasingle.h](#gammagammasingle_h)

[gammagammaleptonpair.h](#gammagammaleptonpair_h)

[inputParameters.h](#inputParameters_h)

[nBodyPhaseSpaceGen.h](#nBodyPhaseSpaceGen_h)

[photonNucleusCrossSection.h](#photonNucleusCrossSection_h)

[upcevent.h](#upcevent_h)

[beam.cpp](#beam_cpp)

[beambeamsystem.cpp](#beambeamsystem_cpp)

[gammaaluminosity.cpp](#gammaaluminosity_cpp)

[gammagammaleptonpair.cpp](#gammagammaleptonpair_cpp)

[gammagammasingle.cpp](#gammagammasingle_cpp)

[incoherentPhotonNucleusLuminosity.cpp](#incoherentPhotonNucleusLuminosity_cpp)

[incoherentVMCrossSection.cpp](#incoherentVMCrossSection_cpp)

[inputParameters.cpp](#inputParameters_cpp)

[narrowResonanceCrossSection.cpp](#narrowResonanceCrossSection_cpp)

[nucleus.cpp](#nucleus_cpp)

[photonNucleusCrossSection.cpp](#photonNucleusCrossSection_cpp)

[readinluminosity.cpp](#readinluminosity_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

[wideResonanceCrossSection.cpp](#wideResonanceCrossSection_cpp)

**Functions**

N/A

starlightdpmjet.h

**Included in files**

[starlight.cpp](#starlight_cpp)

[starlightdpmjet.cpp](#starlightdpmjet_cpp)

**Functions**

starlightDpmJet

init

produceEvent

produceSingleEvent

produceDoubleEvent

setSingleMode

setDoubleMode

setMinGammaEnergy

setMaxGammaEnergy

setProtonMode

starlightlimits.h

**Included in files**

[gammagammaleptonpair.h](#gammagammaleptonpair_h)

[readinluminosity.h](#readinluminosity_h)

[twophotonluminosity.h](#twophotonluminosity_h)

**Functions**

N/A

starlightparticle.h

**Included in files**

[pyhthiadecayer.h](#pythiadecayer_h)

[upcevent.h](#upcevent_h)

[starlightparticle.cpp](#starlightparticle_cpp)

**Functions**

starlightParticle

~starlightParticle

setPdgCode

getPdgCode

setCharge

getCharge

setFirstParent

getFirstParent

setLastParent

getLastParent

setFirstDaughter

getFirstDaughter

setLastDaughter

getLastDaughter

getStatus

setStatus

setVertex

getVertex

starlightparticlecodes.h

**Included in files**

[eventfilewriter.cpp](#eventfilewriter_cpp)

[starlightparticlescodes.cpp](#starlightparticlecodes_cpp)

**Functions**

jetsetToGeant//Converts a jetset code into a GEANT codes

starlightpythia.h

**Included in files**

[starlight.cpp](#starlight_cpp)

[starlightpythia.cpp](#starlightpythia_cpp)

**Functions**

starlightPythia

~starlightPythia

init

produceSingleEvent

produceDoubleEvent

produceEvent

setSingleMode

setDoubleMode

setMinGammaEnergy

setMaxGammaEnergy

setFullEventRecord

starlightStandalone.h

**Included in files**

[main.cpp](#main_cpp)

[starlightStandalone.cpp](#starlightStandalone_cpp)

**Functions**

starlightStandalone

~starlightStandalone

init

run

configFileName

eventDataFileName

setConfigFileName

setEventDataFileName

boostEvent

twophotonluminosity.h

**Included in files**

[starlight.cpp](#starlight_cpp)

[twophotonluminosity.cpp](#twophotonluminosity_cpp)

**Functions**

twoPhotonLuminosity

~twoPhotonLuminosity

twoPhotonDifferentialLuminosity

D2LDMDY

D2LDMDY\_Threaded

integral

radmul

integrand

Nphoton

upcevent.h

**Included in files**

[eventchannel.h](#eventchannel_h)

[filewriter.h](#filewriter_h)

[gammaavm.h](#gammaavm_h)

[pythiadecayer.h](#pythiadecayer_h)

[starlight.h](#starlight_h)

[starlightpythia.h](#starlightpythia_h)

[starlight.cpp](#starlight_cpp)

[upcevent.cpp](#upcevent_cpp)

**Functions**

upcEvent

~upcEvent

addParticle

addVertex

addGamma

getParticles

getVertices

getGammaEnergies

operator=

operator+

boost

vector3.h

**Included in files**

[lorentzvector.h](#lorentzvector_h)

[vector3.cpp](#vector_cpp)

**Functions**

vector3

~vector3

GetVector

SetVector

operator +=

operator =

operator -=

X

Y

Z

Mag2

Mag

operator <<

wideResonanceCrossSection.h

**Included in files**

[gammaavm.cpp](#gammaavm_cpp)

[wideResonanceCrossSection.cpp](#wideResonanceCrossSection_cpp)

**Functions**

wideResonanceCrossSection

~wideResonanceCrossSection

crossSectionCalculation