Contents

Overview:

The eSTARlight Monte Carlo models photon-Pomeron interactions in electron-ion collisions. The physics approach for the photon-Pomeron interactions is described in Lomnitz and Klein, arXiv:1803.06420 (2018).

eSTARlight has several input files, all of which are expected to be in the same directory as the estarlight code. User-specified input parameters are read from a file named "slight.in"; these parameters are described below in Input.

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

What's new/changed:

Although eSTARlight inherits many methods from STARlight, it required sufficient changes to motivate a separate distribution. As such, the source code includes several components from STARlight as well as many changes and new classes. A brief summary of the changes is as follows:

New classes:

- gammaeluminosity: Generates look-up tables for electron-ion collisions.
- e_wideResonanceCrossSection: Coherent vector meson production using wide resonance in eX collisions.
- e_narrowResonanceCrossSection: Coherent vector meson production using narrow resonance in eX collisions.
- e_starlightStandalone: Similar to STARlight case, calls methods to initialize and produce events and decay them.
- e_starlight: Reads in inputParameters and checks their validity.
- e_main: Driver program, initiates and calls e_starlightStandalone.
- eXevent: Contained for e+X event information (analogous to upcEvent in STARlight).

Modified classes

- beambeamsystem: Add support for electron beam.
- readinluminosity: Methods to read the look up tables for _ep_ and _eA_ have been added.
- photonNucleusCrossSection: Methods to calculate photon flux and $\gamma X \rightarrow VX$ cross-section.
- nucleus: Add support for electron (Z=1,A=0).
- inputParameters: Add inputParamters for eSTARlight, including: min(max) γ energy(k) and virtuality(Q^2), number of k and Q^2 bins, electron energy, ..
- .
- gammaavm: Add methods to generate kinematic variables and momenta to finite virtuality. Also generate outgoing electron and target.
- filewriter: Increased precision for event catalogue. Necessary for scattered target and electron.

- eventfilewriter: Add implementation for eXevent to include scattered target and electron in event catalogue.

Installation:

To obtain the latest version:

```
-git clone git@github.com:mlomnitz/eSTARlight.git
```

Alternatively:

-Visit https://estarlight.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 eSTARlight:

```
- First create your build directory <BUILDDIR> (e.g. mkdir bin)
- $ cd <BUILDDIR>
- $ cmake <PathToSource>
- $ make
```

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

To clean the build:

```
- $ make clean
```

To run eSTARlight, a configuration file, slight.in, is needed. Examples of
slight.in may be found in the config/ directory.

To run:

```
$ ./e_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

Input:

The input parameters are listed below with typical values for e-Au collisions at the proposed eRHIC collider. Optional parameters are denoted with *, and Legacy parameter is used to indicate options in the set up file that have been inherited from STARLight but might not be accurately implemented in eSTARlight.

baseFileName	# The name of the output files. eSTARlight will copy the input slight.in to baseFileName.in, and produce output files baseFileName.txt and baseFileName.out. (slight)
BEAM_1_Z = 1	# Charge of beam one projectile. For electron-ion collisions, beam 1 must be the electron/positron (+/-1), at present both are treated equally
BEAM_1_A = 0	# Atomic number of beam one projectile (0 for e).
BEAM_2_Z = 79	# Charge of beam two projectile (Au = 79).
BEAM_2_A = 197	# Atomic number of beam two projectile (Au = 197).
BEAM_1_GAMMA = 35295	# Lorentz boost for beam one projectile(pz>0). These are 18 GeV electrons.
BEAM_2_GAMMA = 106.6	# Lorentz boost for beam two projectile(pz<0). These are 100 GeV/n Au ions.
W_MAX = -1	# Maximum value for the gamma-pomeron center of mass energy, in GeV. Setting W_MAX = -1 tells eSTARlight 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 $2\hbar c\sqrt{\frac{\gamma_1\gamma_2}{R_1R_2}}$. These are defined in src/inputParameters.cpp
W_MIN = -1	#Min value of w. Minimum value for the gamma-pomeron center of mass energy, W, in GeV. Setting W_MIN = -1 tells eSTARlight 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.
W_N_BINS = 40	#Bins W maximum and minimum values for W and the number of w bins in the lookup tables.
RAP_MAX = 8.	# Legacy parameter: Maximum rapidity of produced particle. Mostly left over from STARlight implementation.
RAP_N_BINS = 80	# Legacy parameter: Number of rapidity bins used in the cross section calculation. Mostly left over from STARlight implementation.

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 = 12	# Specifies the production mechanism to be used in event generation. At present eSTARlight includes two options: **PROD_MODE=12:** Coherent photonuclear vector meson production assuming narrow resonances. **PROD_MODE=13:** Coherent photonuclear vector meson production assuming wide resonances. This option should be used for exclusive ρ^0 production.
N_EVENTS = 10	#Number of events produced.
PROD_PID = 443013	# This selects the channel to be produced, in PDG notation. Currently supported options are list below.
RND_SEED = 34533	# Seed for random number generator.
MIN_GAMMA_Q2*	# Specifies whether the user desires to set a minimum value for the photon mass. By default, eSTARlight will set this to physical limit $Q^2_{min} = m_e^2 k^2/E_e(E_e - k)$.
MAX_GAMMA_Q2*	# Specifies whether the user desires to set the maximum value for the photon mass. By default the value is set to $Q^2_{max} = 4E_e(E_e - k)$ with the added requirement that the individual photons satisfy longitudinal coherence $l_c = 2k/(Q^2 + M_V^2)$
INT_GAMMA_Q2_BINS*	# Specifies whether the user desires to change the number of Q^2 bins used when preparing the look-up tables, the default value is 400.
QUANTUM_GLAUBER = 1	# Species whether a quantum or classical Glauber extarpolation is to be used for nuclear targets. 1 = Quantum Glauber, 0 = Classical Glauber.
SELECT_IMPULSE_VM = 0	# Species whether the impulse approximation is to be used. 1 = Use impulse approximation, 0 = don't.
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/GeV^2 + 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)
```

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 " ")
OUTPUT_FORMAT = 0
                       #Used to set the output file format.
                       0 (or not specified):  means the default
                       slight.out format is used.
                       1: 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.
                       2: HEPMC3 format is used to write slight.hepmc
                       file. Requires HepMC3 installation and compilation
                       with -DENABLE_HEPMC3=ON flag
                       3: Lund format is used. Output file is named
                       slight_LUND.txt.
```

Channels of Interest:

Pomeron-Photon Channels

At present only the photon-pomeron channels have been included in eSTARlight (production modes 12 and 13). The channels included are:

jetset id	particle
113	rho0

223	omega
333	phi
443011	J/psi --> e+e-
443013	J/Psi --> mu+mu-
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 eSTARlight 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 eSTARlight. 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 eStarlight 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 eSTARlight

```
$ mkdir <build-dir>
```

and change into it

```
$ cd <build-dir>
```

Run CMake with the option to enable DPMJet

```
$ cmake <path-to-estarlight-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 eSTARlight with DPMJet interface -----------

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

The files needed are:

slight.in (eSTARlight 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:

```
$ ./e_starlight < my.input
```

[DPMJET reads from direct input/interactive]

Output

eSTARlight outputs an ASCII file named slight.out unless otherwise specified by the OUTPUT_FORMAT input. The first 4 lines are used to store some of the important configuration options used to produce the event. The information contained in these lines is as follows:

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 the atomic number of beam 1(2) and beam1(2)LorentzGamma is the Lorentz gamma factor associated to beam 1(2)

These lines are then followed by a brief description of the user settings for the exchanged photons in the collisions, as follows:

PHOTON: nkbins fixedQ2 nQ2bins minQ2 maxQ2

where nkbins is the number of steps (with exponential steps) used for the photon energy look-up tables, fixedQ2 indicates whether the user selected a fixed(=1) range for the photon virtuality or used the physical limits (=0). nQ2bins states the number of bins in Q^2 used for the look-up tables, minQ2 and maxQ2 indicate the minimum and maximum value of Q^2 used to generate events. If the Q^2 range was not selected by the user (i.e. fixedQ2 = 0) minQ2 and maxQ2 are displayed as 0.

These lines are followed by the event catalogue. 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 (eSTARlight does not 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 line describing the kinematics of the photon in the reference frame where the target (p or A) is at rest.

GAMMA: k Q2 ,

where k is the energy of the photon and Q2 is the invariant mass of the virtual photon.

This is followed by information related to the scattered target (X = p or A) which emerges from the collision.

t: event_t ,

where, as expected, event_t is the four momentum transfer squared at the target vertex.

TARGET: px py pz E ,

where px, py and pz are the three vector components of the scattered target three vector and E is it’s energy.

The information related to the scattered target is followed by the scattered electron or source.

SOURCE: px py pz E ,

where, again, px, py and pz are the components of the outgoing electron three vector and E is it’s energy.

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 pz 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.

Analysis

We have also provided a series of macros to facilitate analysis of the output. The next paragraphs will, briefly, describe some of the material included in package:

- <PathToSource>/trunk/utils/ConvertStarlightAsciiToTree.C:
  This ROOT macro can be used to convert the eSTARlight ASCII output file (slight.out) into a ROOT file. The macro stores the simulation set up, 4 vectors for each of the incoming colliding particles photon energy in target frame, photon virtuality, transferred momentum at the target vertex, scattered target 4-momenta, scattered source 4-momenta, and finally the vector meson and decay daughters 4-momenta.
  To run the macro cd to the directory which contains slight.out and run:
  ```
  root –b -q –l <PathToSource>/trunk/utils/ConvertStarlightAsciiToTree.C
  ```
- <PathToSource>/analysis/:
  This directory contains some template files illustrating how to read out the output from ConvertStarlightAsciiToTree.C to produce histograms for a given analysis. The files myHist.cxx and myHist.h define a C++ class to initiate, fill and plot the individual histograms for your analysis. e_AnalyzeTree.cxx and e_AnalyzeTree.h pass the desired variables from a given event to the histograms in the myHist class. e_AnaTree.C is the main program and iterates over the individual events. Finally, we have provided a script to compile the objects and call them within the ROOT environment. To run an analysis simply type:
  ```
  sh e_run.sh file_to_study.root
  ```

Production

In the directory <PathToSource>/production/ we have included a series of Jupyter-Notebooks illustrating how eSTARlight can be used to for a sample of different studies. These notebooks are self documented, with information on each step. Included with this release are (cd <PathToSource>/production/):

- ./templates: This directory contains a series of template input files (i.e. slight.in) used for the different studies.
- ./event_generation/Event-Generation.ipynb: This notebook automates the steps needed to run the full pipeline, i.e.: generate sample in eSTARlight, convert the ASCII file to ROOT tree and run an analysis on the ROOT file.
- ./Au_vs_Fe_test/Gold_Iron_Q2_test.ipynb: This notebook can be used to calculate the cross-section in eA for gold and iron

targets as a function of the photon Q^2 and produce the plot of the ratio scaled by $A^{4/3}$.

- ./accel_x_secs/x_section_calculations.ipynb: This notebook calculates the cross-sections and rates at the different accelerators (EIC, JLEIC and LHeC) for different vector meson species and outputs tables to be inserted in a LaTeX document.