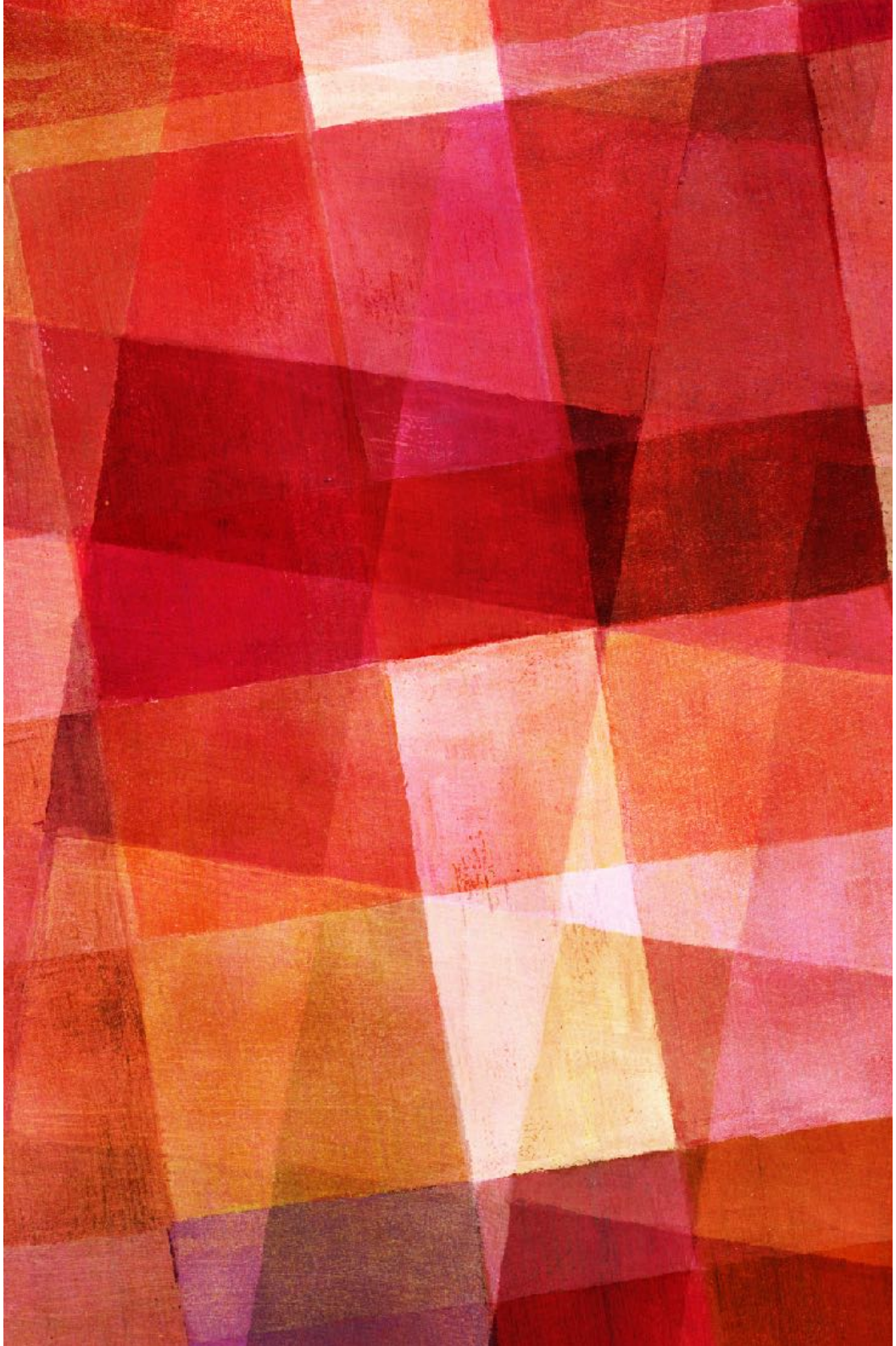
MONTE CARLO EVENT GENERATION WITH RADIATIVE QED PROCESSES IN DEEP-INELASTIC SCATTERING

Nicolas Pierre, for the COMPASS collaboration

March 31, 2017 – DPG'17 HK 57.3



CONTENTS

- Radiative corrections : characterization and impact.
- The DJANGOH generator for radiative events
- Inclusion of Radiative Corrections in Monte-Carlo

INTRODUCTION TO RADIATIVE CORRECTION

When extracting PDFs or FFs, obtain cross-sections :

$$\sigma_{\exp} = \sigma_{th}\left[F_n\left(x, Q^2, \ldots\right)\right]$$

Taking into account higher order corrections :

$$\sigma_{th} = \sigma^{(0)}\left[F_n\right] + \alpha_{em}\sigma^{(1)}\left[F_n\right] + \ldots$$

Experimental problem :

can't distinguish radiative photon from non-radiative ones…

OBSERVED CROSS-SECTION

It is the convolution of the true cross-section times the radiator function :

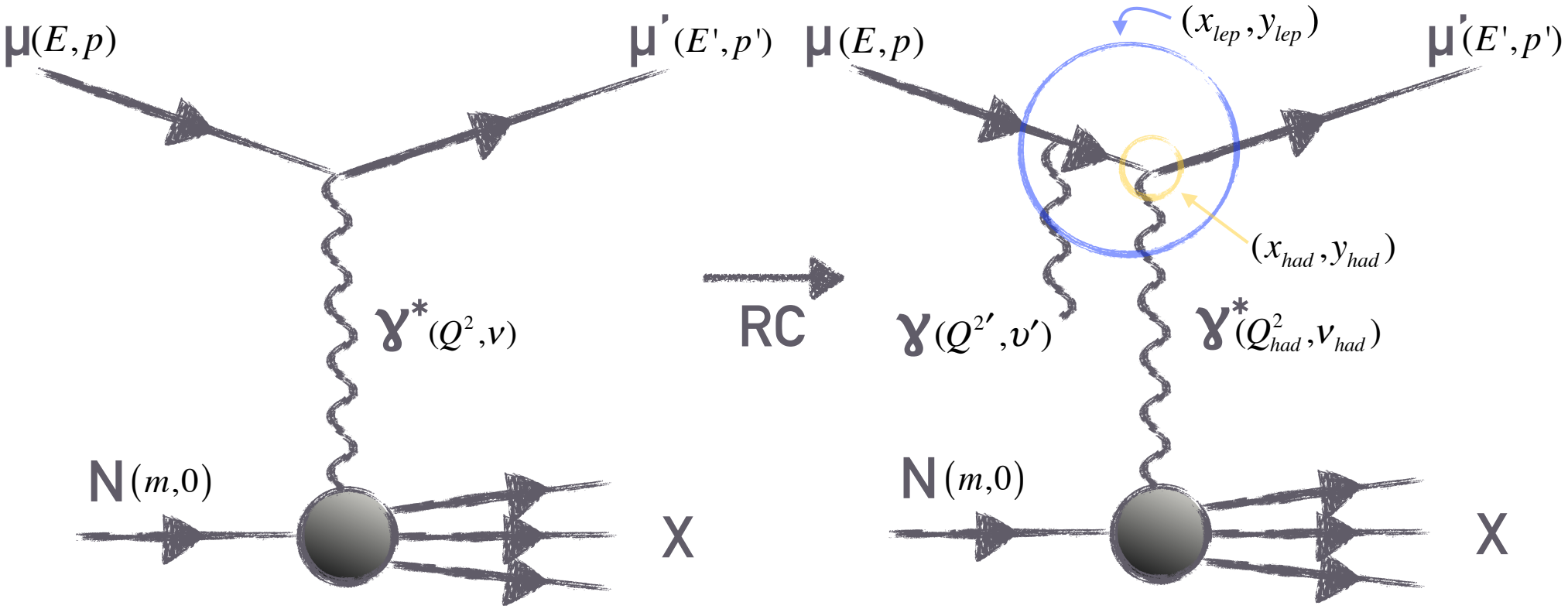
$$d\sigma^{obs}(p,q) = \int \frac{d^3k}{2k^0} R(l,l',k) d\sigma^{true}(p,-q,k)$$

Same goes for structure functions :

$$F_n^{obs}(x,Q^2) = \int d\tilde{x}\, d\tilde{Q}^2 R_n(x,Q^2,\tilde{x},\tilde{Q}^2) F_n^{true}(\tilde{x},\tilde{Q}^2)$$

Formulas can be extended for higher-order multi-photon emission.

CHARACTERIZATION OF A RADIATIVE EVENT



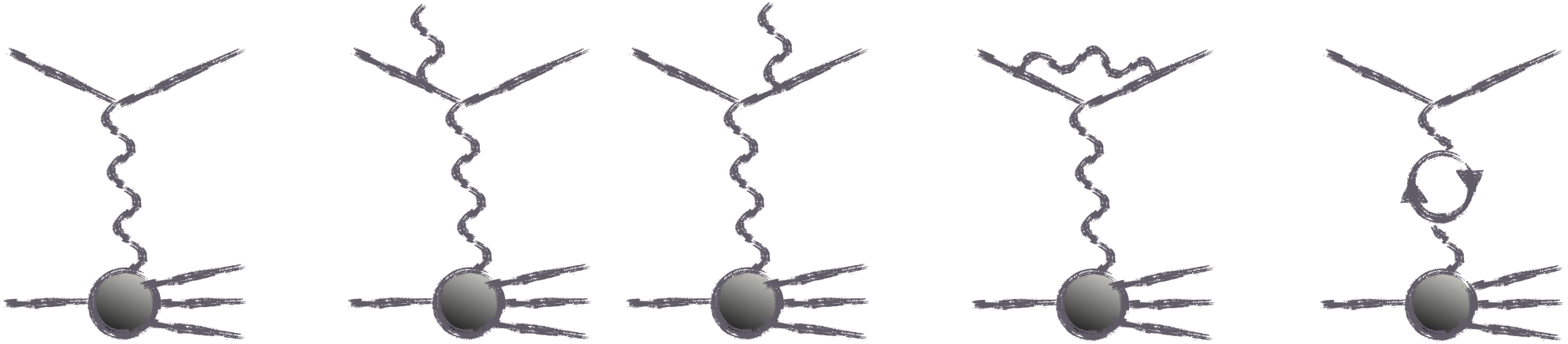
SIDIS = Semi-Inclusive DIS (obs. of one hadron of the final state)

$$x \overset{lab}{=} \frac{Q^2}{2M\upsilon} \qquad y \overset{lab}{=} \frac{E - E'}{E}$$

Radiative event
=
Event containing a real radiated photon

IMPACT OF RADIATIVE CORRECTIONS

The radiative correction is defined by : $\eta(x,y) = \dfrac{\sigma_{Born}(x,y)}{\sigma_{total}(x,y)}$



Born | Internal Bremsstrahlung | Vertex Correction | Vacuum Polarization

Impact in SIDIS :

Difference between the hadronic and leptonic kinematic variables induces that eventually some hadrons fall into the wrong (x,y) bin.

Application of the correction factor to multiplicities is performing kinematic event migration, 'redirecting' these hadrons.

RC @ COMPASS

For hadron radiative correction, use of TERAD, which does second order calculation of these corrections.

- However, still don't know where the radiative photon goes…

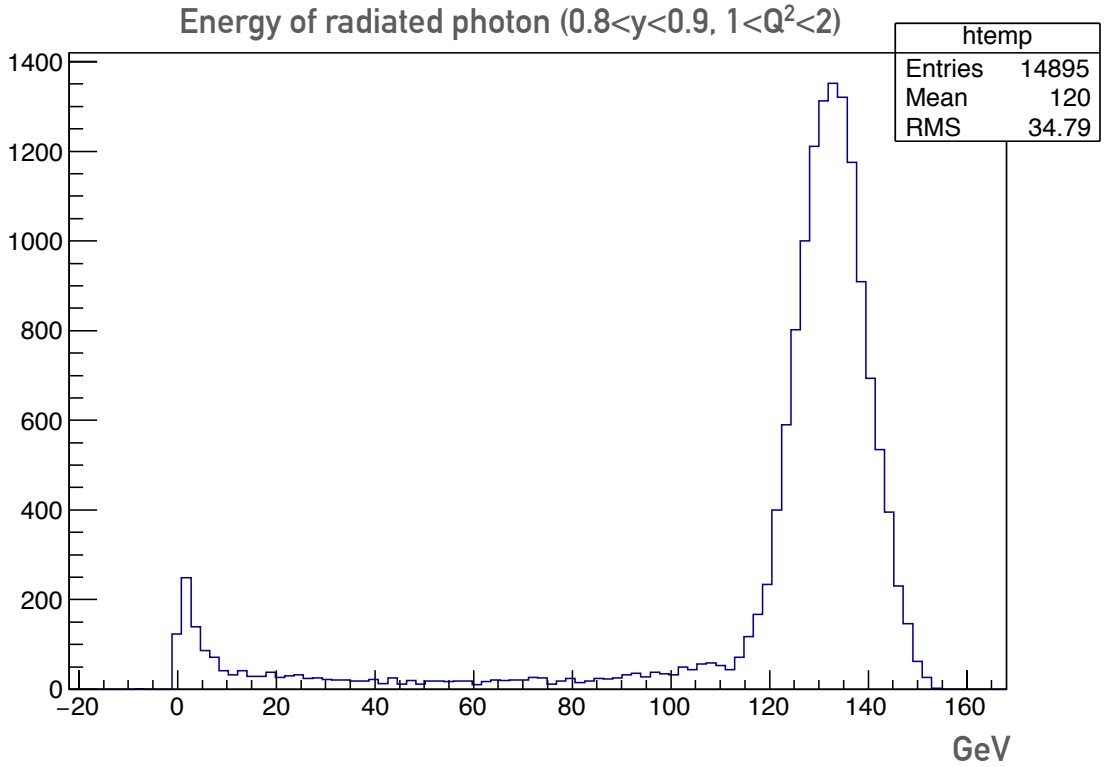
For this, need a radiative event generator.
COMPASS used RADGEN generator in previous analyses.

FINDING THE BEST RC GENERATOR

RADGEN : produces more hard photons (high energy photons) than soft photon (low energy photons).

Naïve thinking : in theory, more soft than hard photons.

But : MC simulation + RADGEN do not describe well data : too much hard photon.



Can we use another MC generator for radiative events and see if it reproduces the results of RADGEN ?

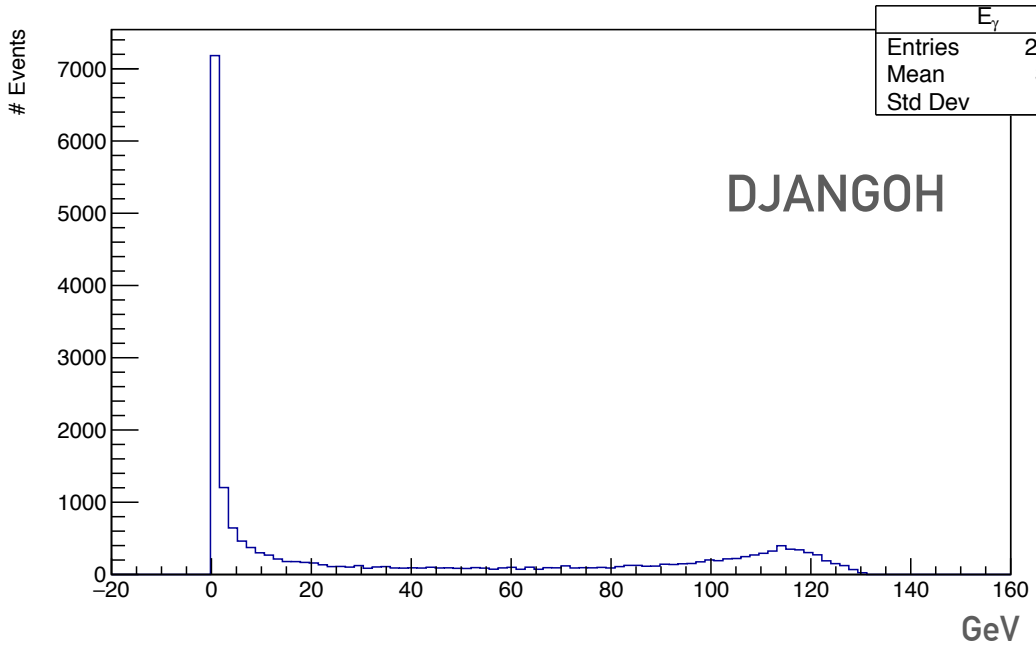
DJANGOH

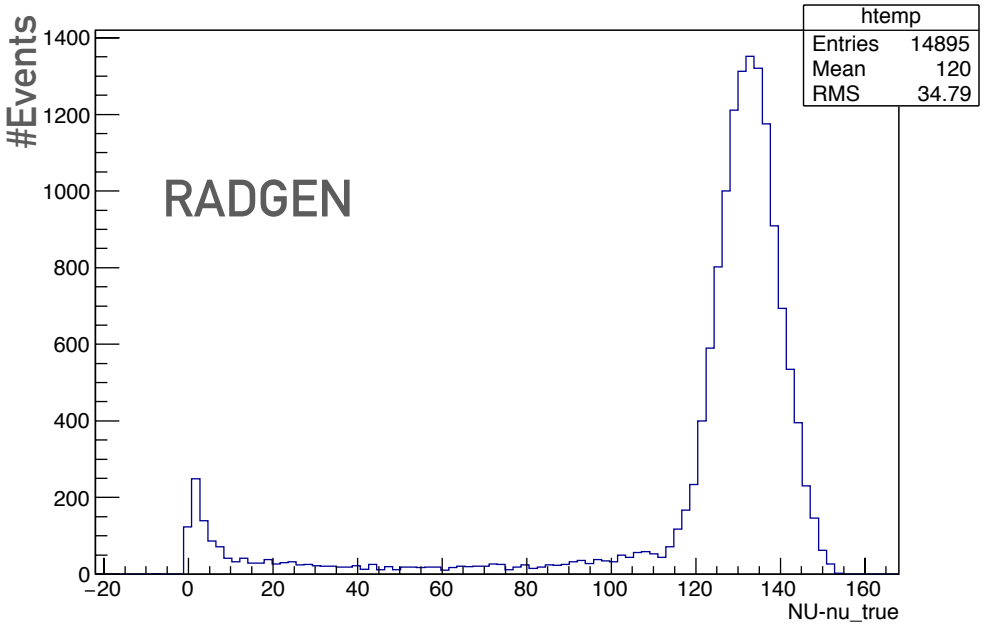
DJANGOH, concatenation of DJANGO and HERACLES :

- Event generator for neutral/charged current ep interactions at HERA by H. Spiesberger.
- Simulates DIS including both QED and QCD radiative effect.
- Includes single photon emission from lepton/quark line, self energy corrections and complete set of one-loop weak corrections.
- Includes also the background from $ep \rightarrow ep\gamma$.
- Capable of obtaining hadronic final state via the use of JETSET.
- Modified to work for μp interactions.
- Uses exact calculations and no approximations.
- Fortran framework.

DJANGOH/RADGEN COMPARISON

Energy of radiated photon ($0.8<y<0.9$, $1<Q^2<2$)

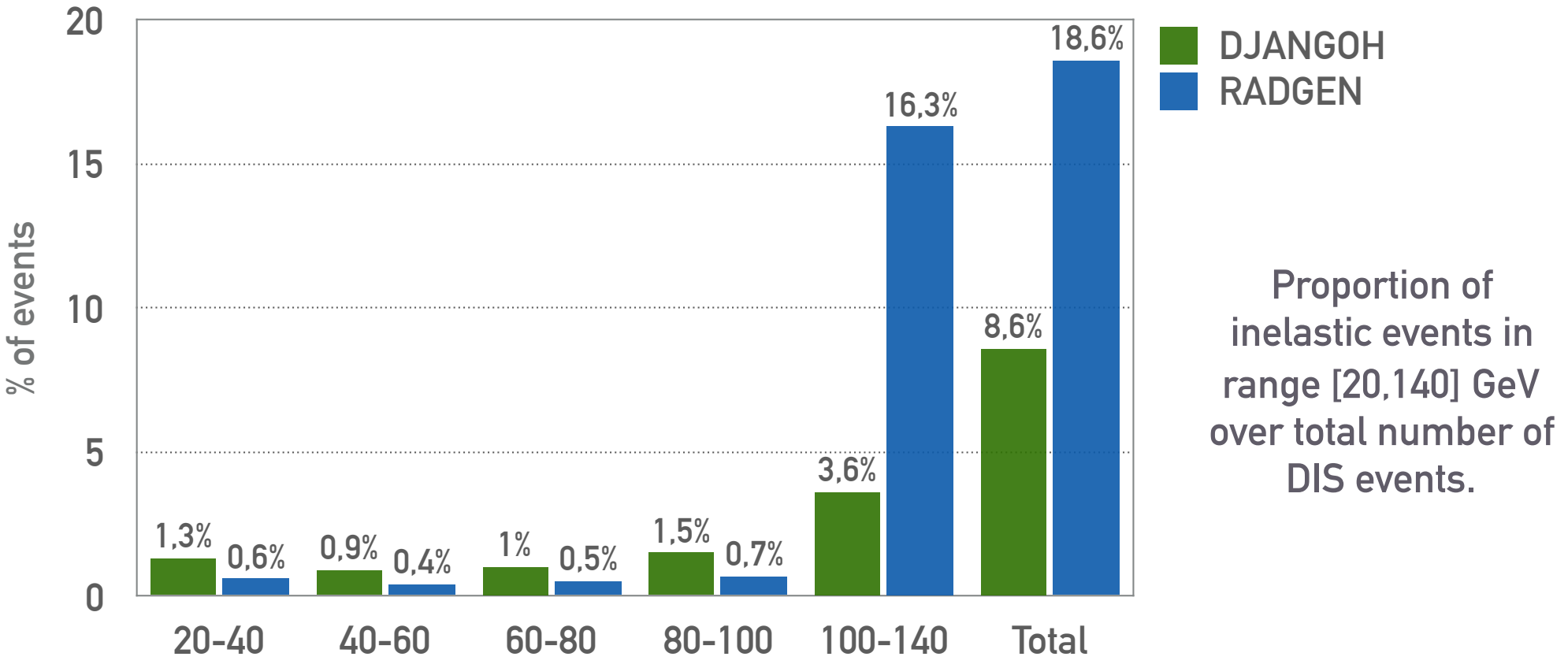




First observation :

- DJANGOH produces more soft photons than hard photons

DJANGOH/RADGEN COMPARISON



Proportion of inelastic events in range [20,140] GeV over total number of DIS events.

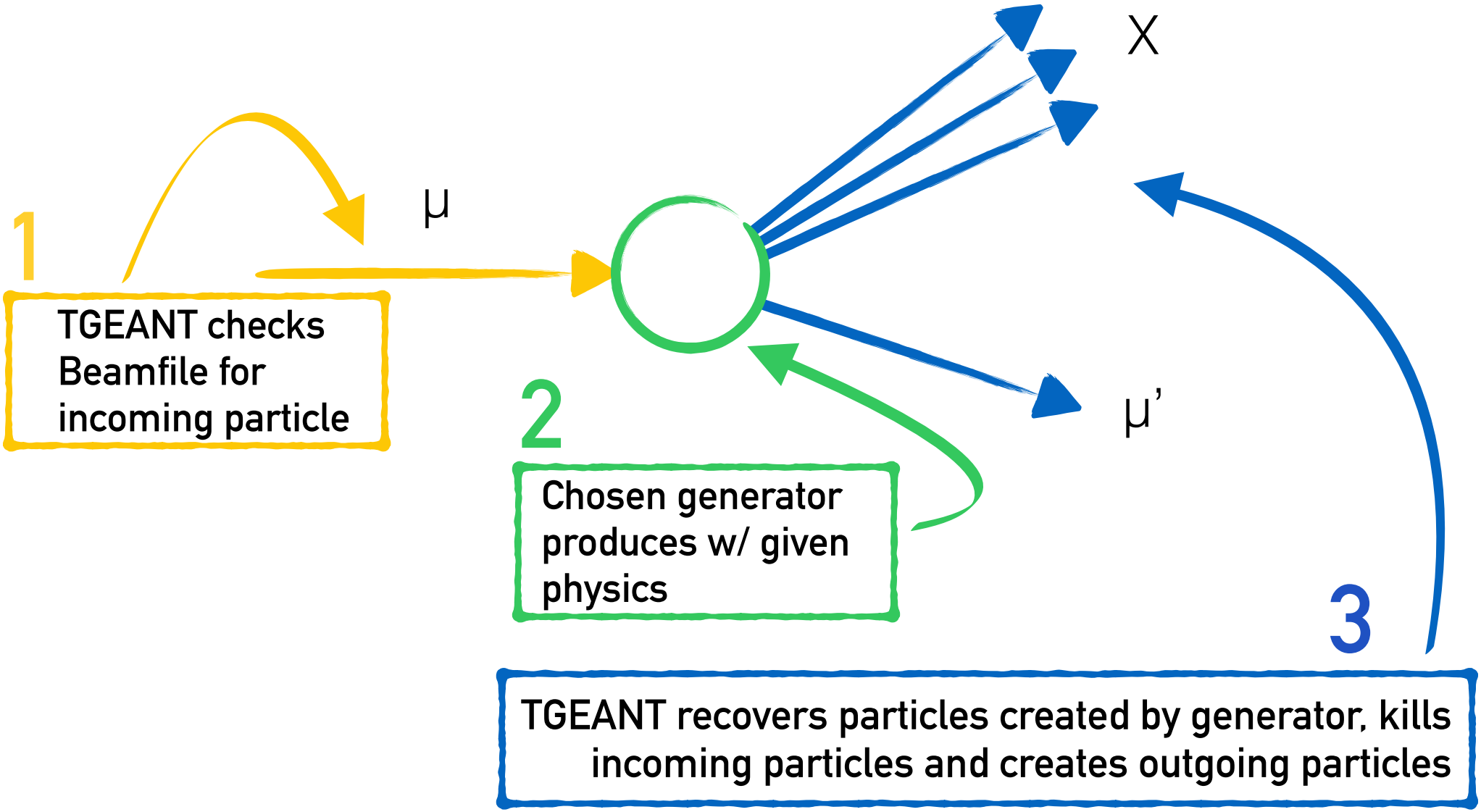
DJANGOH produces less hard photons than RADGEN

TGEANT

« A GEANT4-based Monte-Carlo simulation for the COMPASS-II experiment.

- C++ framework
- Modular
- Can handle event generators for specific interaction simulation

EVENT GENERATION HANDLING IN TGEANT



INTERNAL IMPLEMENTATION OF DJANGOH

TGEANT : Beam Gen

TGEANT

C++ Interface to Djangoh recovers infos of Beamfile from TGEANT

Interface run Djangoh as subroutine

Interface send LUJETS result of Djangoh to TGEANT

DJANGOH

Rotation of 4-V of outgoing particles

TGEANT creates outgoing particles, kill beam

Recover results in detectors

TGEANT

INTERNAL IMPLEMENTATION OF DJANGOH

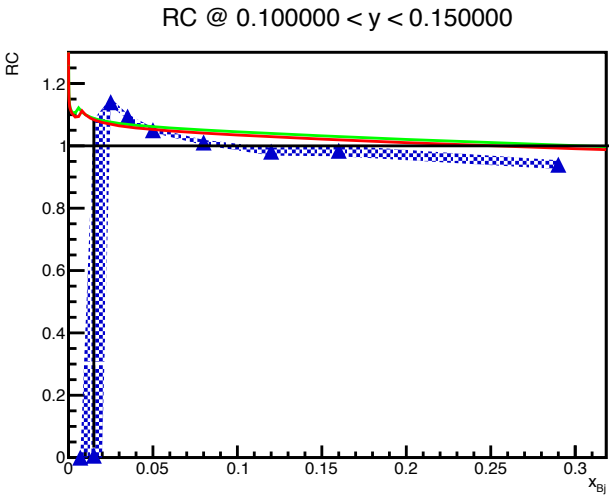
Interface Class : TDjangoh

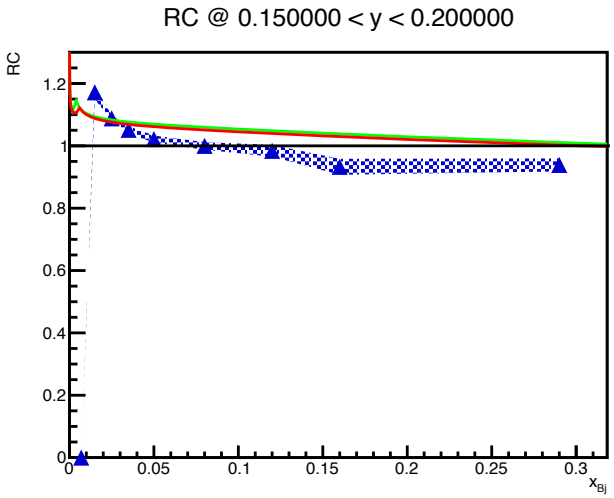
- Is a standalone class (only ROOT dependent)
- Creates instance of Djangoh that can be manipulated in any C/C++ environment
- TDjangoh can be used within the ROOT framework (future use ?)

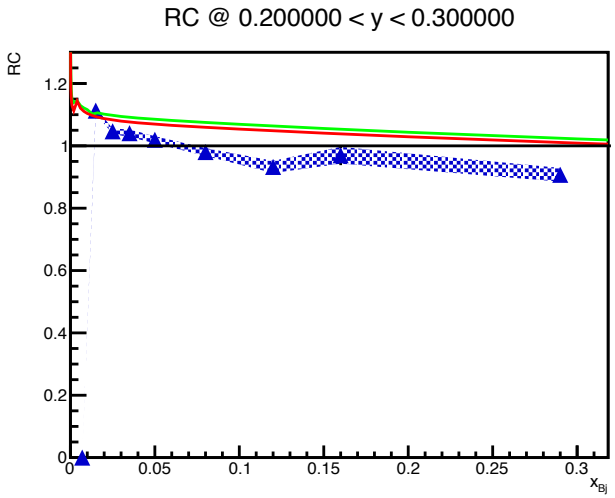
Process Class : T4DjangohProcess

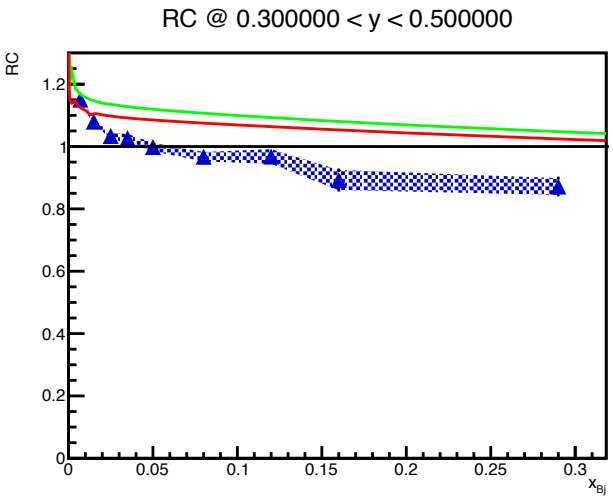
- Is a TGEANT class
- Manipulates instance of TDjangoh
- Do the I/O transfer of TGeant to TDjangoh

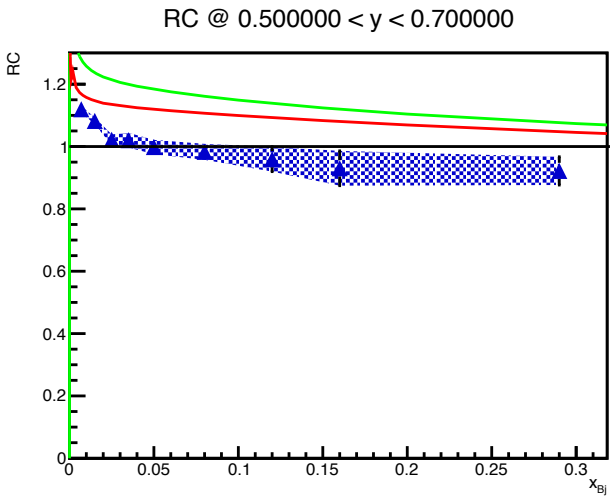
INCLUSIVE RC FACTOR IN (X,Y) BINS











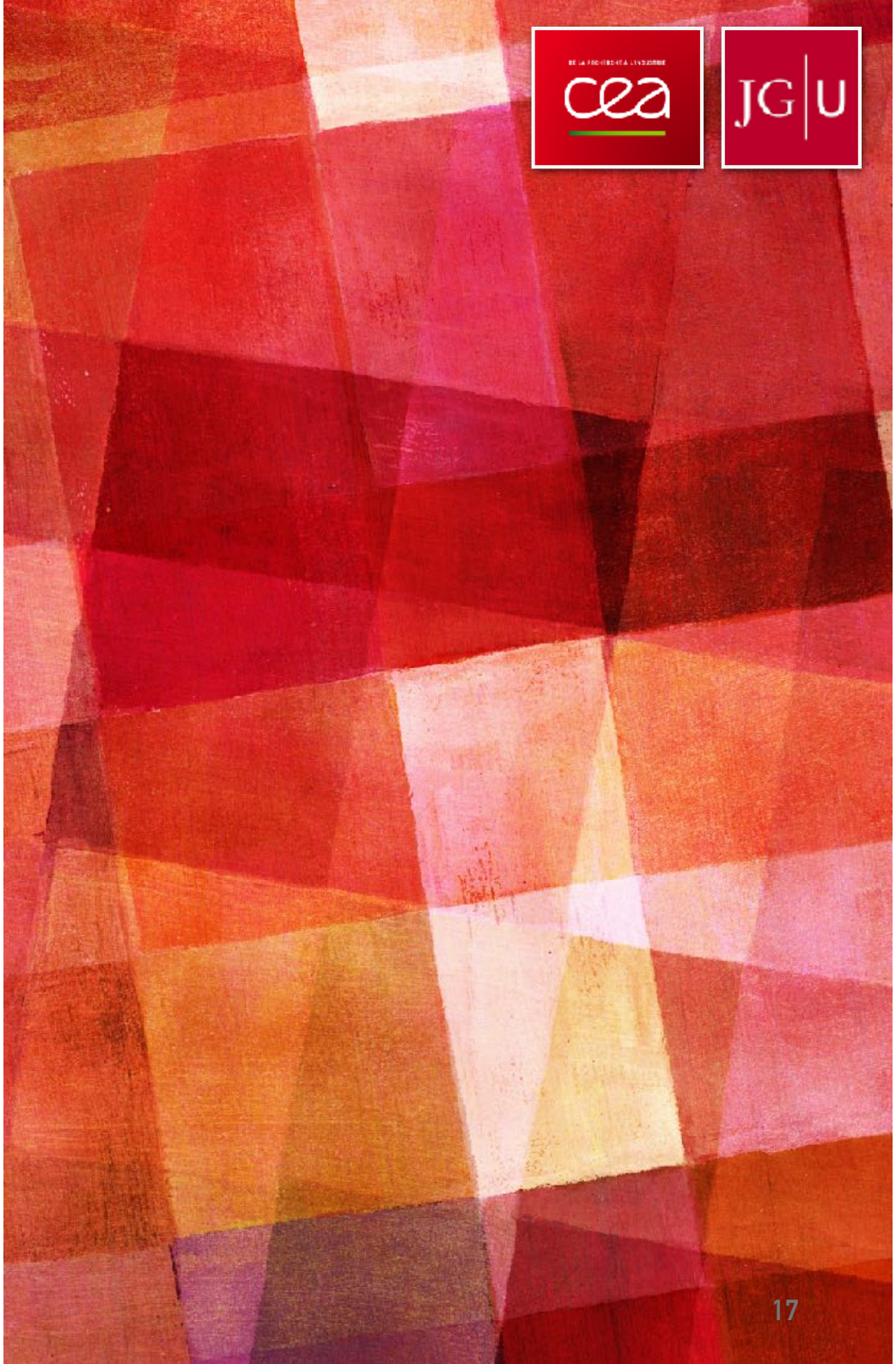
Discrepancy between DJANGOH and TERAD : what is the cause ?

One trail : difference in structure functions parametrizations

Correction up to 17%, mean correction around 5%

CONCLUSION & PROSPECTS

- Radiative Corrections can be calculated exactly thanks to objects like radiator function.
- RC needed for SIDIS cross-sections. Impact of these corrections in this channel are not negligible : corrections goes up to 17%, average at ~5%.
- Idea is to choose the best event generator consistent with real data and use modularity of MC chain to put it in.
- Production of MC with Djangoh as event generator has begun for 2016 setup, hopefully giving better agreement than Radgen. Actually in the reconstruction step.



BACKUP

UNFOLDING

Determination of the true cross-sections from the measured ones :

$$d\sigma^{obs}(p,q) = \int \frac{d^3k}{2k^0} R(l,l',k)\, d\sigma^{true}(p,-q,k)$$

Typical answer : an iterative solution !

But, ill defined :
- no unique solution
- large uncertainties
- numerically unstable

However, with partial functioning : $R(l,l',k) = \frac{I}{k\cdot l} + \frac{F}{k\cdot l'} + \frac{C}{\tilde{Q}^2}$

- Initial state radiation : k.l small for $\sphericalangle(l_{in},\gamma) \to 0$
- Final state radiation : k.l' small for $\sphericalangle(l_{out},\gamma) \to 0$
- Compton peak : Q^2 small for $p_T(l_{out}) \sim p_T(\gamma)$

SOME WORDS ON HADRONIC RADIATION

Cancels with loops, collinear emission give rise to correction of type :

$$\frac{\alpha}{2\pi}\log\left[m_q^2\right] \quad where \quad m_q = 0$$

Solution is to factorize and absorb the divergences into PDF.

$$d\sigma = \sum_f d\hat{\sigma}_f \left[1+\delta_f\left(Q^2, m_q^2\right)\right] q_f\left(x\right) = \sum_f d\hat{\sigma}_f \hat{q}_f\left(x, Q^2\right)$$

However, due to the difference of charge between quarks, there's an isospin violating effect.

INTERFACE CLASS

Changed input method of Djangoh :

- Standard method ⟶ input file.
- But : input file is not efficient when producing 1M events changing input between each generation.
- Solution : drawing correspondence between struct in C++ and COMMON blocks in Fortran.
- Defined input values necessary for Djangoh in Interface.

Struct in C++

```
60  extern "C" struct ihscut
61  {
62      float ixmin;
63      float ixmax;
64      float iq2min;
65      float iq2max;
66      float iymin;
67      float iymax;
68      float iwmin;
69  } ihscut_;
```

Corresponding COMMON block in Fortran

```
51      COMMON /IHSCUT/ IXMIN,IXMAX,IQ2MIN,IQ2MAX,IYMIN,IYMAX,IWMIN
52      REAL            IXMIN,IXMAX,IQ2MIN,IQ2MAX,IYMIN,IYMAX,IWMIN
```

When a value is given to a member of the block in C++, we retrieve the same value in its Fortran counterpart and vice-versa.

INTERFACE CLASS

Output of Interface :

- Lujets_t object containing the list of all particles.
- Lujets_t linked to LUJETS COMMON block.
- Can be obtained from Interface via GetLujets() method.
- Checked consistency between what obtained in Fortran subroutine and what recovered in Interface

```
13  struct Lujets_t
14  {
15    int    N;
16    int    NPAD;
17    int    K[5][4000];
18    double P[5][4000];
19    double V[5][4000];
20  };
```

From Djangoh (fortran)

Event listing (summary)

I	particle/jet	KS	KF	orig	p_x	p_y	p_z	E	m
1	mu-	1	13	0	-1.371	-0.842	115.003	115.015	0.106
2	K-	1	-321	0	1.639	0.142	17.036	17.123	0.494
3	p+	1	2212	0	0.416	-0.413	0.512	1.219	0.938
4	gamma	1	22	0	0.185	0.384	7.085	7.097	0.000
5	gamma	1	22	0	0.017	-0.007	0.170	0.171	0.000
6	pi+	1	211	0	-0.396	-0.119	4.098	4.121	0.140
7	gamma	1	22	0	-0.105	-0.023	1.058	1.063	0.000
8	gamma	1	22	0	-0.109	0.114	2.146	2.151	0.000
9	gamma	1	22	0	-0.050	0.348	3.881	3.897	0.000
10	gamma	1	22	0	-0.085	0.082	1.170	1.176	0.000
11	gamma	1	22	0	-0.116	0.203	3.625	3.632	0.000
12	gamma	1	22	0	-0.085	0.158	1.452	1.463	0.000
13	gamma	1	22	0	0.137	-0.068	1.079	1.090	0.000
14	gamma	1	22	0	0.001	-0.102	0.916	0.922	0.000
15	gamma	1	22	0	0.014	0.111	0.353	0.371	0.000
16	gamma	1	22	0	-0.092	0.033	0.416	0.427	0.000
		sum:	0.00		-0.000	-0.000	160.000	160.938	17.353

From Interface

```
Event Generation..
 RANLUX LUXURY LEVEL SET BY RLUXGO : 4      P= 389
 RANLUX INITIALIZED BY RLUXGO FROM SEEDS      2244          0          0
Event Generated !

Total number of particles : 16
Xbj : 0.0427037 y : 0.281158 Q2 : 3.60494
```

Results from Fortran subroutine are well passed to the Interface

OPTIMIZING TDJANGOH

Djangoh was designed :

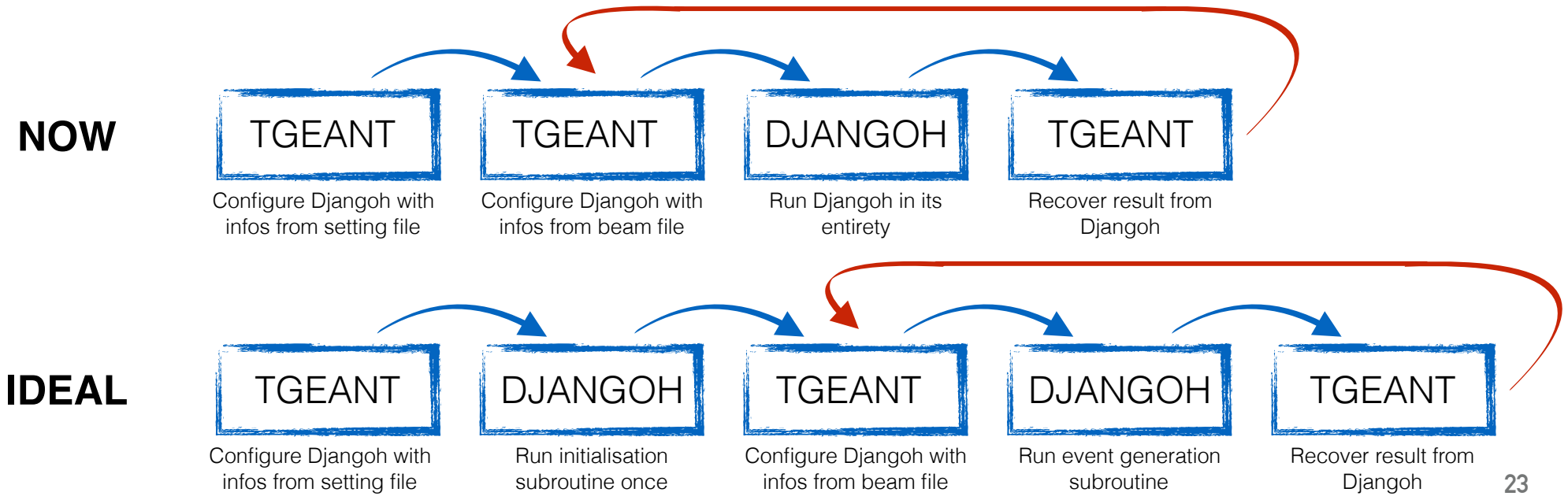
- to work with only one input of incoming lepton energy
- to then run multiple event based on this unique energy

TGEANT needs :

- to generate events with incoming lepton energy picked from beam file
  - means 1 event = 1 energy

The way it is handled :

- ‘open’ Djangoh, run 1 event, ‘close’
- Not really optimized : each time, lot of redundant stuff are made



LOOKING FOR TIME CONSUMING PROCESSES

For 20 events
- Time elapsed : 4m 02s
- Means : 12.1 s/evt

Where does it take so long ?
- Integration step where cross sections for radiative processes are calculated

Is it possible to do it once ?
- Integration step depends on the incoming particle energy, hence not trivial to do it once.

Look at 1 event

	Total time elapsed (s)
Initialisation	0
Integration/Init. for event sampling	14
Sampling	14
Django6	14
Init user routine	14
Event generation	14
End	14

FROM ONE TO A GRID OF CROSS-SECTIONS

For one input energy

COMMON /HSSNC2/
@ E_0

For several input energies

COMMON /HSSNC2/
@ E_0

COMMON /HSSNC2/
@ E_1

⋮

COMMON /HSSNC2/
@ E_n

MODIFICATION OF DJANGOH

Before

SUBROUTINE HSMAIN()

Initialisation step

Receive input from Interface

X-section calculation step

For one energy calculates corresponding X-sections for radiative processes

Event generation step

Generate events according to calculated X-sections for radiative processes

After

SUBROUTINE HSINIT()

Receive input from Interface

For a GIVEN RANGE of energy calculates corresponding X-sections GRID for radiative processes

SUBROUTINE HSEVTG()

Generate events according to calculated X-sections for radiative processes picked in the GRID wrt. INPUT ENERGY

THE CROSS-SECTION GRID

The actual container / The type mimic

```
78  C---                    NREG2N=NBIN2**NDIM2
79        LOGICAL LGLO2,LLOC2
80        double precision ntot2
81        COMMON /HSSNC2/ SIG2,SIG2E,T2GGMA,T2GMAX(NREG2N),
82       +                XX2(50,2),
83       +                FFGO2,DNCG2,FFLO2,DNCL2,GOLD2,
84       +                NM2(NREG2N),NDO2,
85       +                NTOT2,NCAL2,NCA12,NCA22,IBIM2,JCOR2,
86       +                LGLO2,LLOC2
```

```
1   MODULE xSectionModule
2
3     USE ISO_C_BINDING
4
5     TYPE, BIND(C) :: xSection2
6       REAL :: SIG2
7       REAL :: SIG2E
8       REAL :: T2GGMA
9       REAL :: T2GMAX(2500)
10      REAL :: XX2(50,2)
11      REAL :: FFGO2
12      REAL :: DNCG2
13      REAL :: FFLO2
14      REAL :: DNCL2
15      REAL :: GOLD2
16      REAL :: NM2(2500)
17      REAL :: NDO2
18      DOUBLE PRECISION :: NTOT2
19      REAL :: NCAL2
20      REAL :: NCA12
21      REAL :: NCA22
22      REAL :: IBIM2
23      REAL :: JCOR2
24      LOGICAL :: LGLO2
25      LOGICAL :: LLOC2
26    END TYPE
```

The type mimic is placed in a Module in order to be used and recognised as the same type in every in every subroutine (because fortran)

THE CROSS-SECTION GRID

```
220  C-------------------------
221        TYPE(xSection2) :: HSXNC2
222        TYPE(xSection2C) :: HSXCC2
223        TYPE(xSection2E) :: HSXEL2
224        TYPE(xSection31) :: HSXN31
225        TYPE(xSection32) :: HSXN32
226        TYPE(xSection33) :: HSXN33
227        TYPE(xSection34) :: HSXN34
228        TYPE(xSection31C) :: HSXC31
229        TYPE(xSection32C) :: HSXC32
230        TYPE(xSection33C) :: HSXC33
231        TYPE(xSection31E) :: HSXE31
232        TYPE(xSection32E) :: HSXE32
233        TYPE(xSection33E) :: HSXE33
234        COMMON /HSXSEC/ HSXNC2(100),HSXCC2(100),HSXEL2(100),
235       +                HSXN31(100),HSXN32(100),HSXN33(100),HSXN34(100),
236       +                HSXC31(100),HSXC32(100),HSXC33(100),
237       +                HSXE31(100),HSXE32(100),HSXE33(100)
238        COMMON /HSGRID/ GDSIZE, GDMEAN, GDSDDV
239  C-------------------------
```

Then declaration of COMMON Block that consists of tables of types

=

COMMON Block of GRIDS

THE CROSS-SECTION GRID

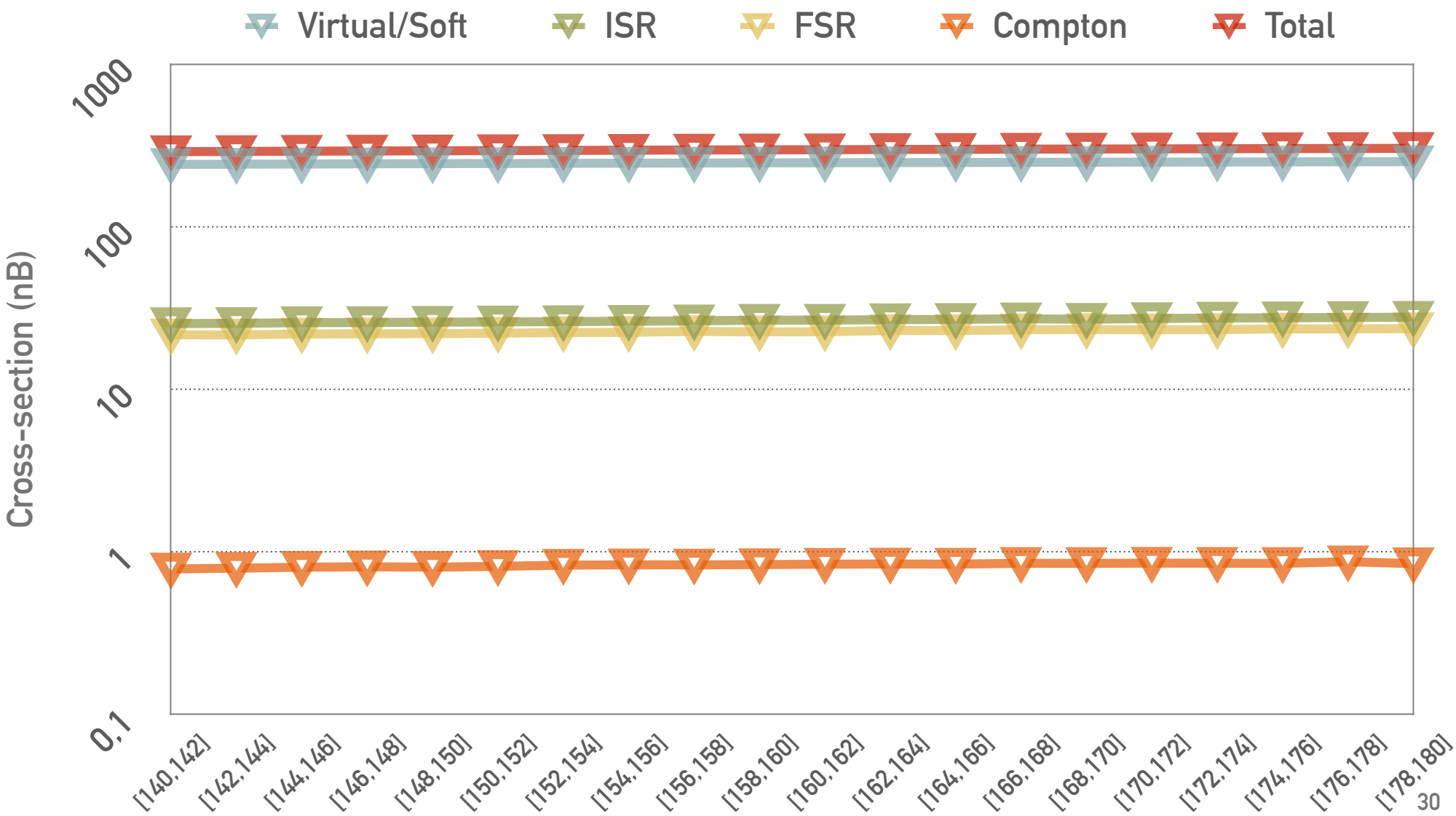
```
1157  C*************************************************************************
1158  C                INTEGRATION / INITIALIZATION FOR EVENT SAMPLING
1159  C                DETERMINATION OF GLOBAL/LOCAL MAXIMA
1160  C*************************************************************************
1161
1162        GDSIZE=4
1163        GDSDDV=4.0
1164        GDSCLE=2.0
1165        GDMEAN=160.0
1166  C---LOOP-LOOP OVER XSECTION GRID
1167        DO 3980 I = 1, GDSIZE
1168           INFOSA=0
1169           EELE=GDMEAN-GDSDDV+(I-1)*GDSCLE
1170           WRITE(455,*)'EELE VALUE : '
1171           WRITE(455,*) EELE
1172  C---NEUTRAL CURRENT
1173           WRITE(455,*)'NEUTRAL CURRENT TREE'
1174  C---BORN TERM + SOFT & VIRTUAL CORRECTIONS-------------------------
1175  C
1176           IF(INT2(1).GE.1) THEN
1177              WRITE(LUNOUT,'(///,5X,2A)') 'START INTEGRATION FOR ',
1178        *          CHNAME(1)
1179  C---INTEGRATION
1180              INCCC=1
1181              CALL HSINIT(INCCC,EPS0,NBIN2,NDO2,SIG2,
1182        *                 SIG2E,XX2)
1183              HSXNC2(I)%SIG2=SIG2
1184              HSXNC2(I)%SIG2E=SIG2E
1185              WRITE(LUNOUT,'(//A/5X,A/5X,1PE12.4,A,1PE12.4,A)')
1186        *          'CROSS SECTION (WITH ERROR ESTIMATE) FOR THE CHANNEL:',
1187        *          CHNAME(1), HSXNC2(I)%SIG2, ' +/- ', HSXNC2(I)%SIG2E,
1188        *          ' NB'
1189              WRITE(LUNTES,'(5X,A,1PD10.1)') ' RELATIVE ACCURACY REQUIRED:',
1190        &          EPS0
```

Integral Calculation for all X-Sections

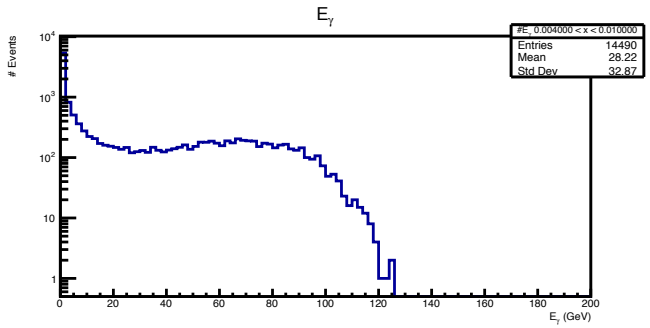
To initialize the grid, just loop over the different bins and take the center of the bin as energy of incoming lepton for cross-section calculation.

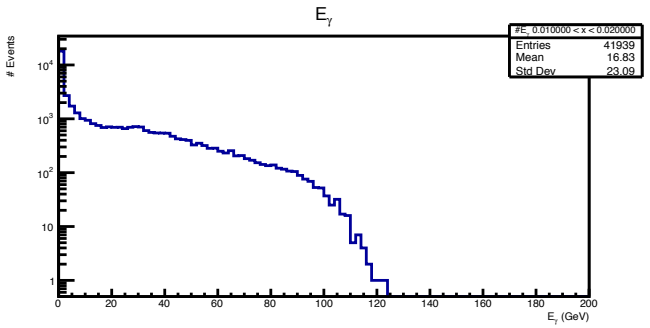
VARIATION OF CROSS-SECTION WITH ENERGY

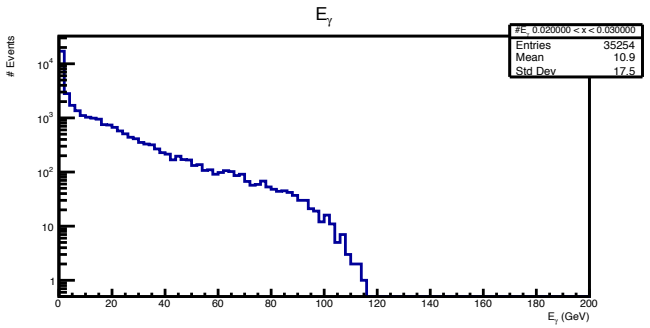
Diff. on [140,180] : *9.5 nB* *2.38 nB* *2.0 nB* *0.06 nB* *13.9 nB*

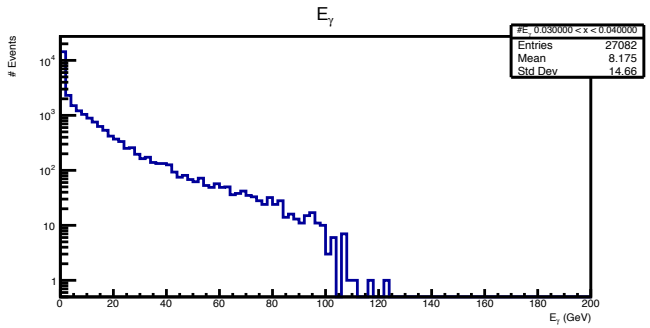


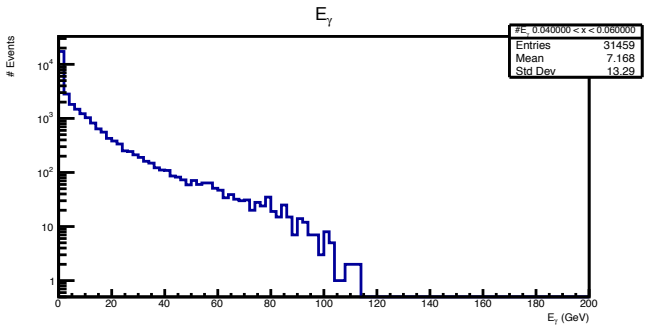
VARIATION OF PHOTON ENERGY WITH X

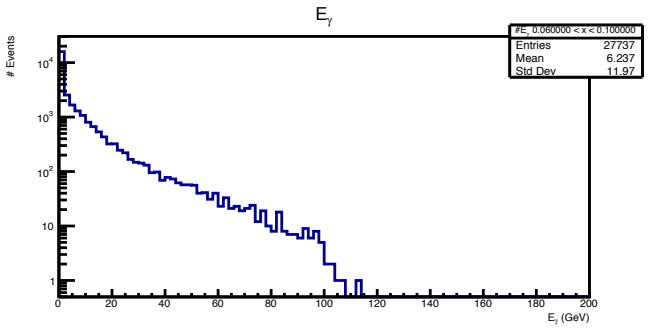


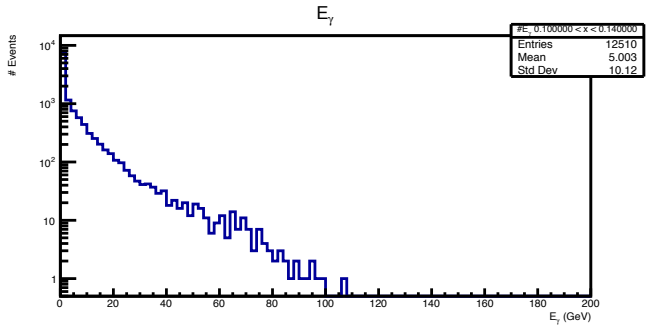


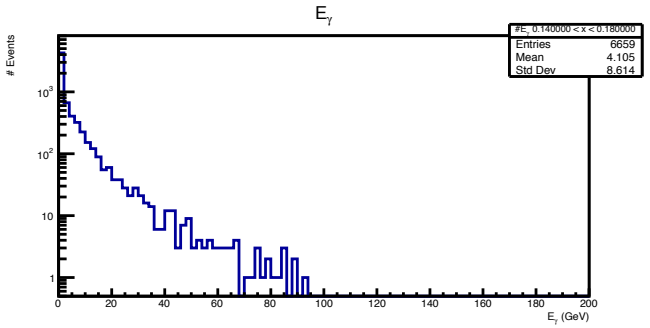


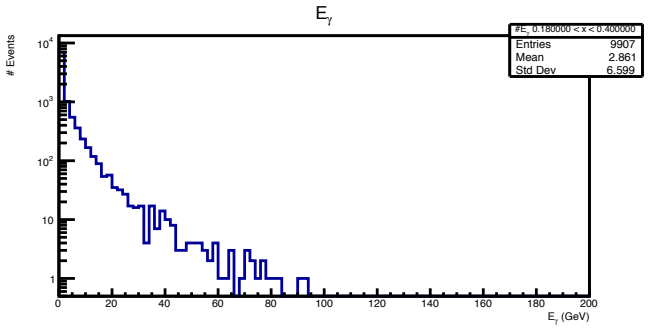












VARIATION OF PHOTON ENERGY WITH Y