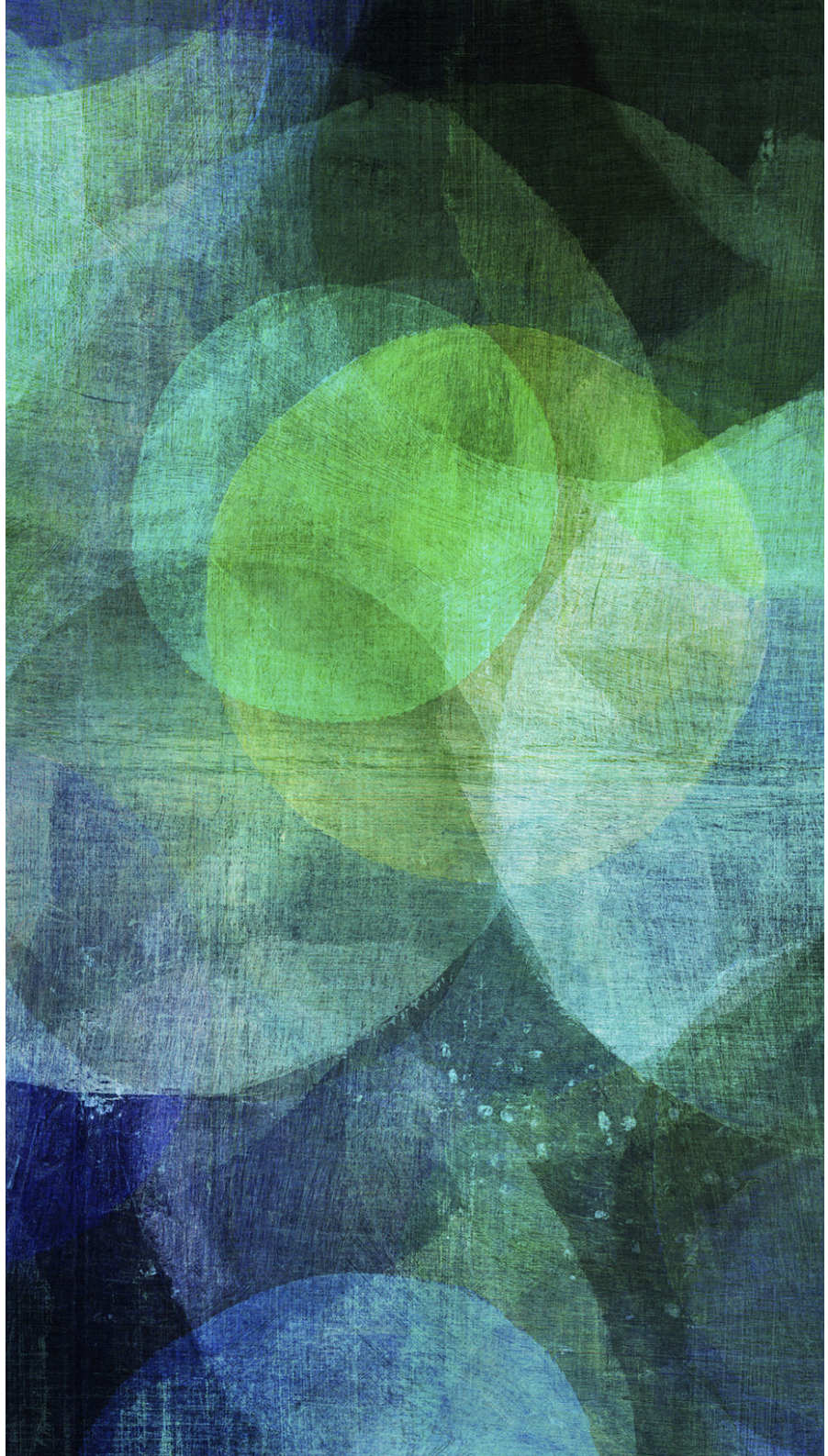
RENORMALISATION AND IMPLEMENTATION OF RADIATIVE CORRECTIONS WITH DJANGOH

N. PIERRE, KPH Mainz

08.03.17



(BRIEF) SUMMARY

- Renormalization, regularization
- Radiative corrections
- Djangoh
- TGEANTxTDJANGOH

RENORMALIZATION : WHAT IS IT ?

« Renormalization is a collection of techniques in Quantum Field Theory used to treat infinities coming from calculated quantities by altering the values of these quantities to compensate for effects in self-interactions.

DIVERGENCE IN QED

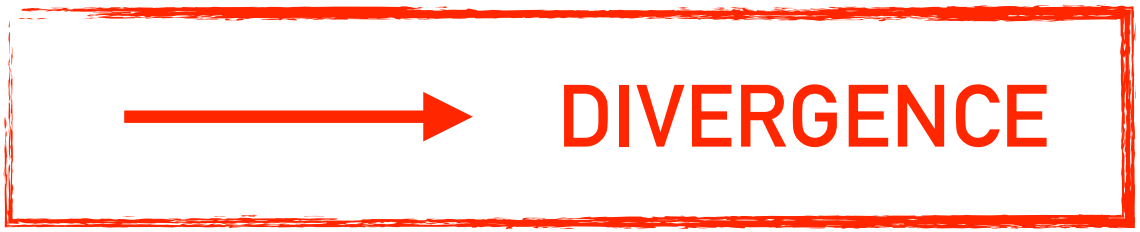
30's : Discovery of divergence in perturbative calculations.
(Born, Heisenberg, Jordan, Dirac)

30's/40's : Descriptions of these divergences (e.g. virtual particle closed loop)
(Stueckelberg, Schwinger, Feynman, Tomonaga, Dyson)

Virtual particles : obey E/p conservation but can be off-shell.

If there's a loop, all variation of loop particle must be compensated within the loop.

To find amplitude of the loop, one must integrate over all possible combinations !

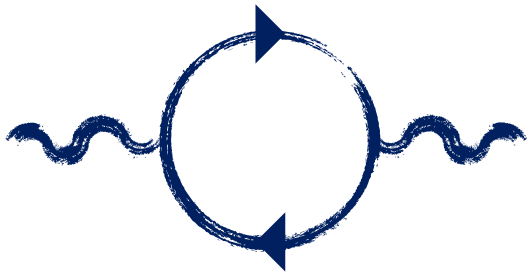


UV DIVERGENCE

The UV divergence is defined by :

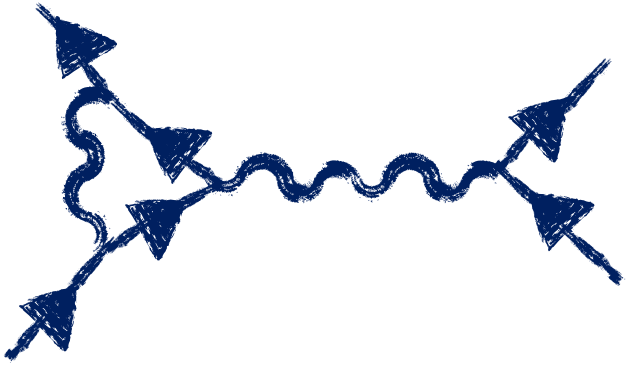
- a region of integral with loop particles with high E/p
- short wavelength, high frequency in field
- short proper time between emission and absorption

There are three one-loop divergent loop diagrams :



Vacuum polarization

Self-energy



Vertex correction

The three divergences correspond to three parameters of the theory :

- field normalisation Z
- mass of the electron
- charge of the electron

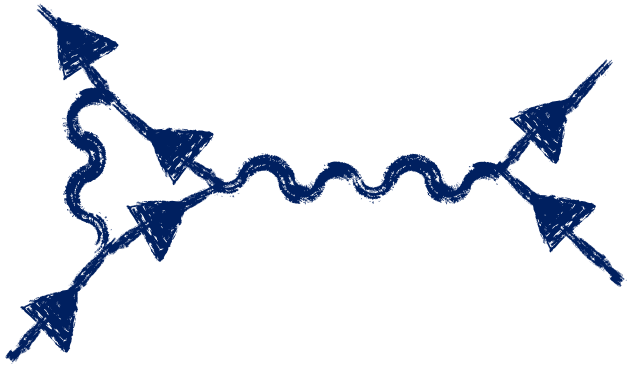
IR DIVERGENCE

The IR divergence is due to massless particles.

- charged particles emits coherent photons with infinite wavelength
- amplitude for emitting any finite nb of photon is null.

In this case, don't need to renormalize a parameter of the theory.

Removal of IR divergence by including diagrams similar to vertex correction :

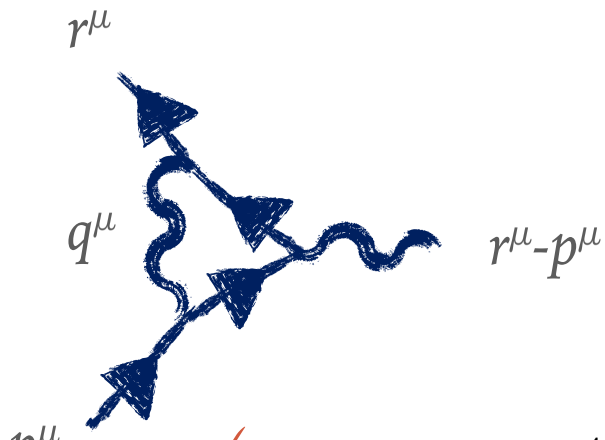


Vertex correction

Two on-shell photon w/ infinite wavelength (similar to bremsstrahlung)

Can't tell difference between VC and Bremsstrahlung : both must be included.

HOW TO DEAL WITH LOOP DIVERGENCE ?



q^μ not only defined uniquely by E/p conservation :

Integration

$$-ie^3\int\frac{d^4q}{(2\pi)^4}\gamma^\mu\frac{i\left(\gamma^\alpha(r-q)_\alpha+m\right)}{(r-q)^2-m^2+i\varepsilon}\gamma^\rho\frac{i\left(\gamma^\beta(p-q)_\beta\right)+m}{(p-q)^2-m^2+i\varepsilon}\gamma^\upsilon\frac{-ig_{\mu\nu}}{q^2+i\varepsilon}$$

At large q^μ : $$e^3\gamma^\mu\gamma^\alpha\gamma^\rho\gamma^\beta\gamma_\mu\int\frac{d^4q}{(2\pi)^4}\frac{q_\alpha q_\beta}{(r-q)^2(p-q)^2q^2}$$ Divergent

Remove this divergence with cutoff : $$e^3\gamma^\mu\gamma^\alpha\gamma^\rho\gamma^\beta\gamma_\mu\int_0^\Lambda\frac{d^4q}{(2\pi)^4}\frac{q_\alpha q_\beta}{(r-q)^2(p-q)^2q^2}$$

SOME WORDS ON REGULARIZATION

Dealing with infinite and divergence is a tricky business.

For ex, what to do with some ∞-∞ quantities ?

Answer : Regularization ! (ad-hoc proc.)

Ex :

$$V(r) = \frac{k}{r}$$

Newton potentiel in sph. coordinates

Singularity in 0.

$$V_\varepsilon(r) = \frac{k}{\sqrt{r^2 + \varepsilon^2}}$$

Introduction of family of one parameter

$$V_\varepsilon(0) = \frac{k}{\varepsilon} < +\infty$$

Defined expression at r = 0

SOME WORDS ON REGULARIZATION

In QFT, modification on the boundaries of integration : introduction of regulator in form of a cutoff such as integral converge.

Putting cutoff to infinity recovers the original integral.

With cutoff, infinite quantities goes finite and can cancel. After cancellation, cutoff is put to infinity and physics results are recovered.

Ex : Zeta function regularization

$$\zeta(s)=\sum_{n=1}^{\infty} n^{-s}=1^{-s}+2^{-s}+3^{-s}+\ldots \xrightarrow{\textit{analytic prolongation}} \zeta(-1)=-\frac{1}{12}$$

Similarly : $$I(n,\Lambda)=\int_0^{\Lambda} p^n\,dp \sim 1+2^n+\ldots+\Lambda^n \to \zeta(-n) \quad \textit{when} \quad \Lambda \to \infty$$

Improving this into an algorithm (Hartle, Garcia, Elizalde) :

$$I(n,\Lambda)=\frac{n}{2}I(n-1,\Lambda)+\zeta(-n)-\sum_{r=1}^{\infty}\frac{B_{2r}}{(2r)!}\frac{\Gamma(n+1)}{\Gamma(n-2r+2)}(n-2r+1)I(n-2r,\Lambda)$$

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 :

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

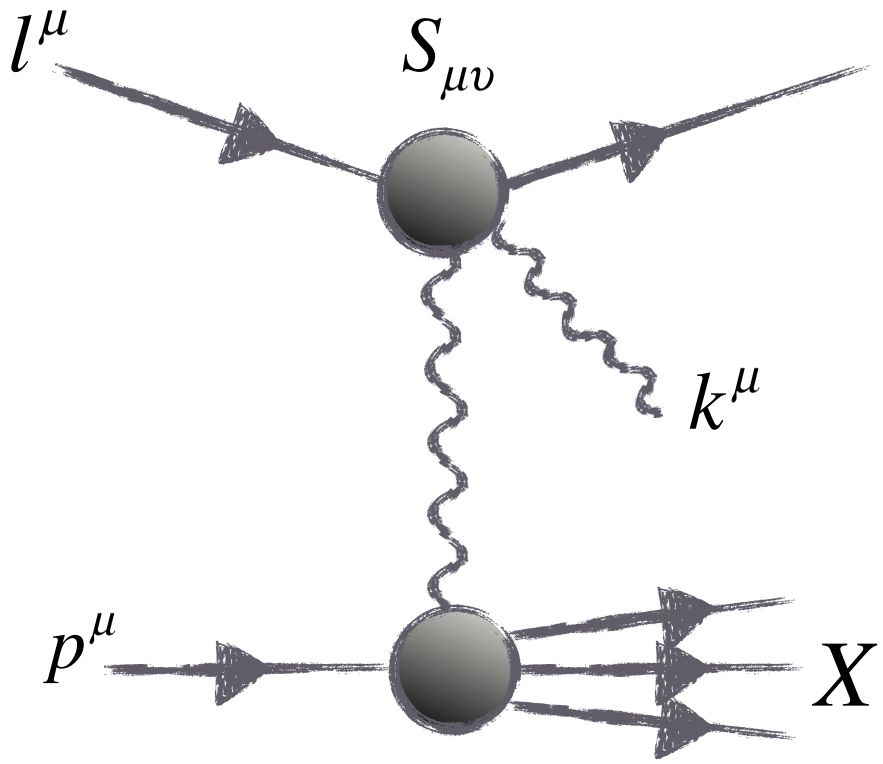
Experimental problem :

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

INTRODUCTION TO RADIATIVE CORRECTION

$O(\alpha)$ corrections are :

- lepton radiation (model ind.)
- hadronic radiation (parton model)
- Interference of lep/had radiation (2 photon exchange)
- Vacuum polarization
- Weak corrections



A single tensor is regrouping all these infos :

the radiative leptonic tensor $S_{\mu\nu}(l,l',k)$

$S_{\mu\nu}$ is :

- gauge invariant
- infrared finite
- universal
- including Born + loop @ $\delta^{(4)}k^\mu$

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.

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 REMARKS ON LEPTONIC RADIATION

$$E^2_{\gamma,\max} \propto Q^2 \frac{1-x}{x}$$

- Large radiation at large Q^2-small x
- Radiation suppressed at small Q^2-large x
- Large negative correction from unconcealed virtual contribution

$$\tilde{Q}^2_{\min} \propto \frac{x^2}{1-x} M_N^2$$

- $\tilde{Q}^2 \ll Q^2$ possible

SOME WORDS ON HADRONIC RADIATION

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

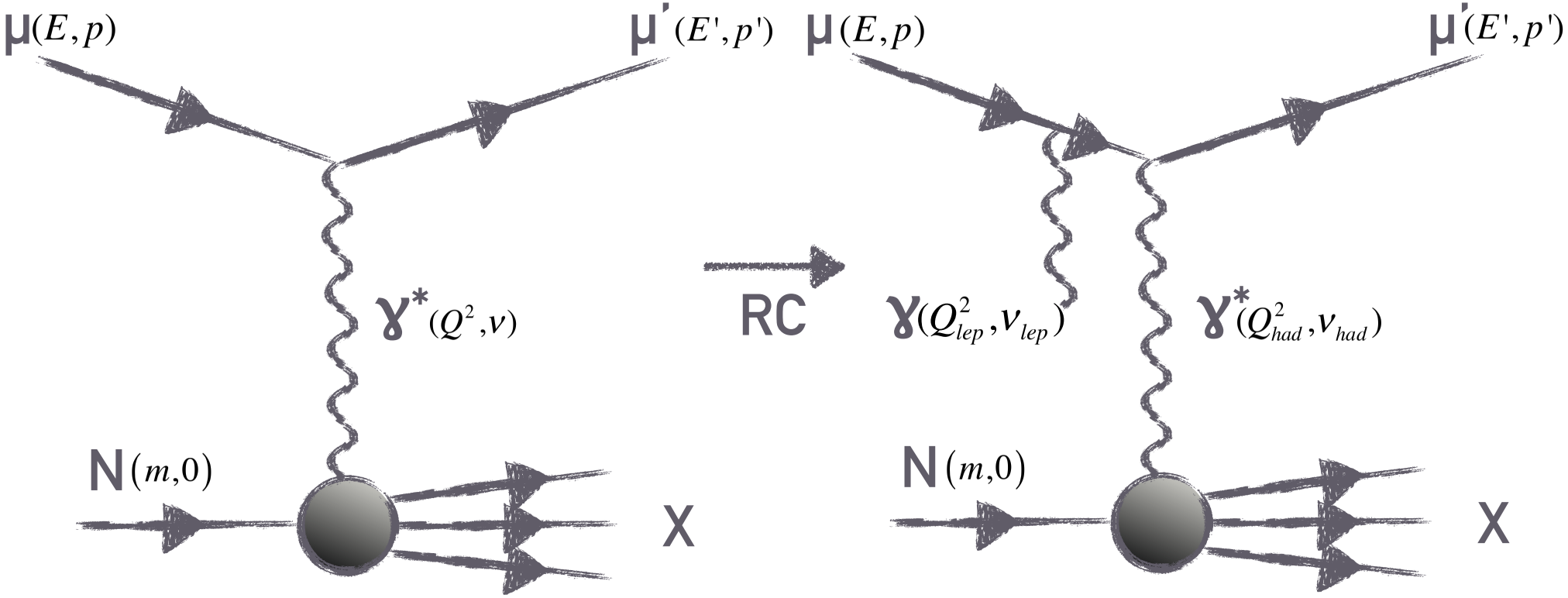
$$\frac{\alpha}{2\pi}\log\left[m_q^2\right] \quad where \quad m_q = 0$$

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.

CHARACTERIZATION OF A RADIATIVE EVENT



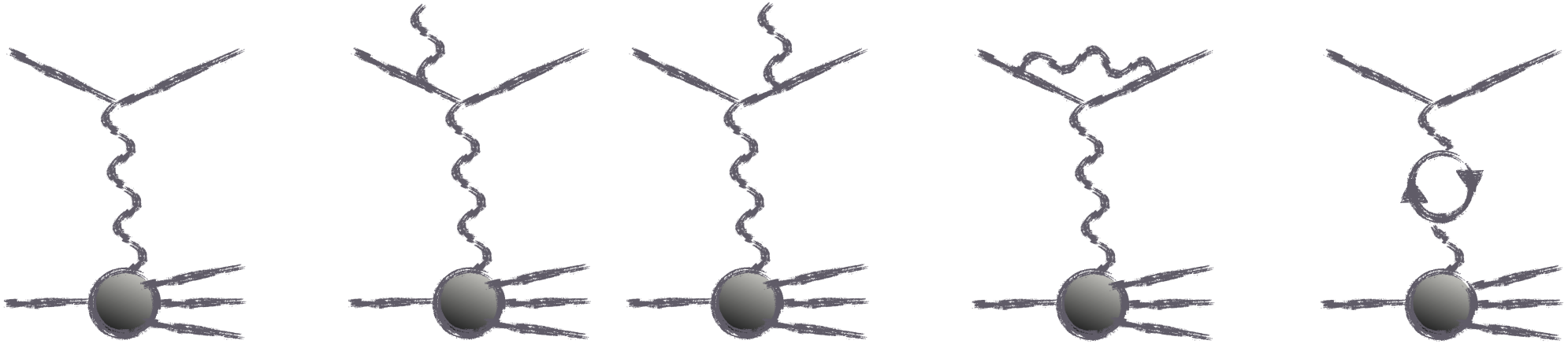
$$\nu_{had} = \nu - E_\gamma$$

$$Q^2_{had} = Q^2 + 2E_\gamma\left(\nu - \sqrt{\nu^2 + Q^2}\cos\theta_\gamma\right)$$

Radiative event
=
Event containing a real radiated photon

IMPACT OF RADIATIVE CORRECTIONS

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

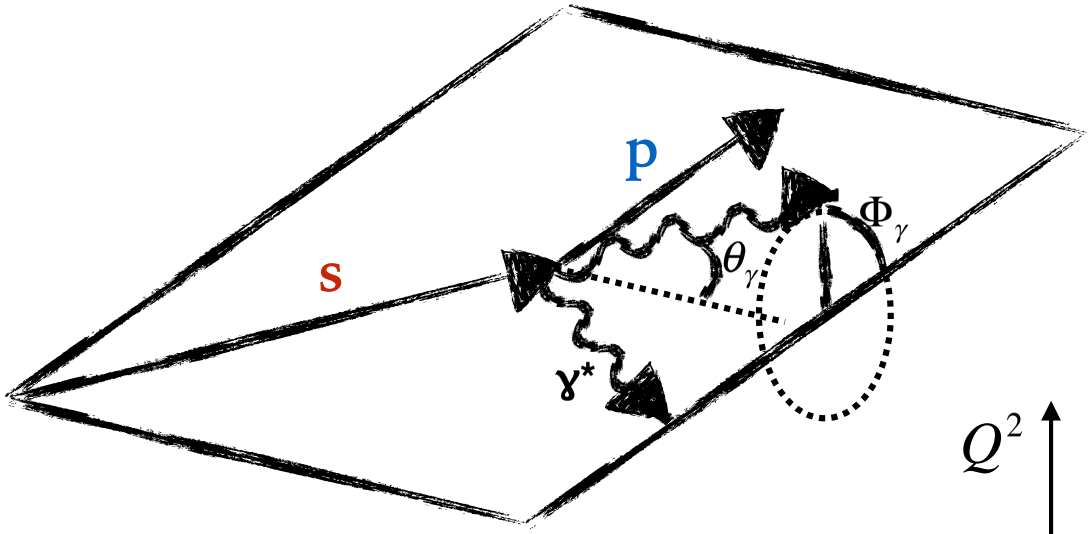


Born | Internal Bremsstrahlung | Vertex Correction | Vacuum Polarization

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

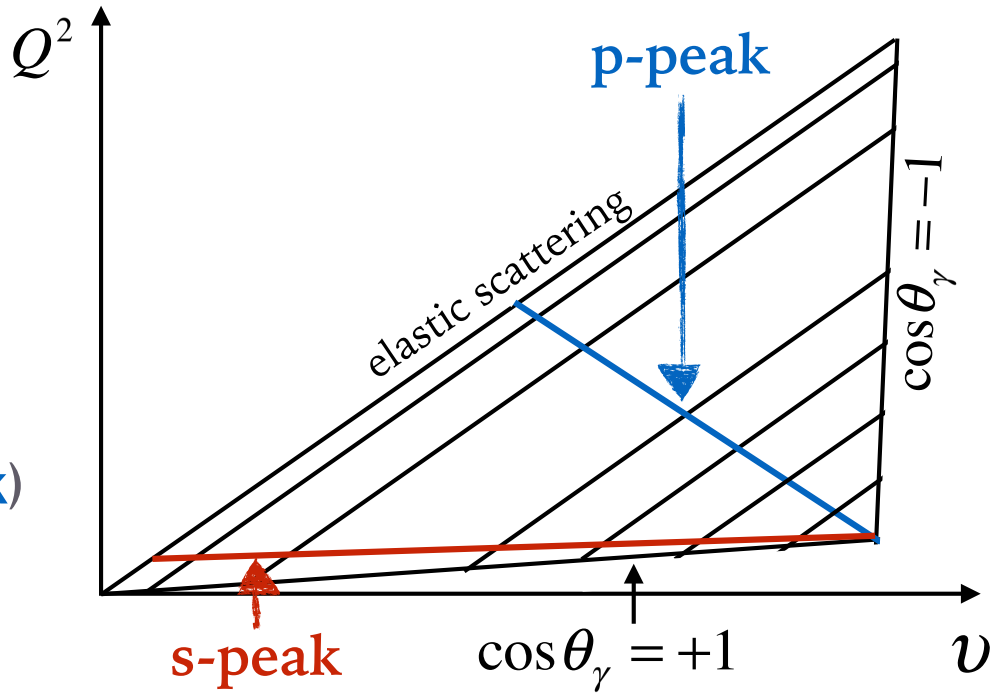
Applying the correction factor to multiplicities is indirectly 'redirecting' hadrons fallen in wrong (x,y) bins into the right ones.

ABOUT EMISSION OF RADIATIVE PHOTONS



θ_γ : polar angle

Φ_γ : azimuthal angle



Two privileged angles of emission :

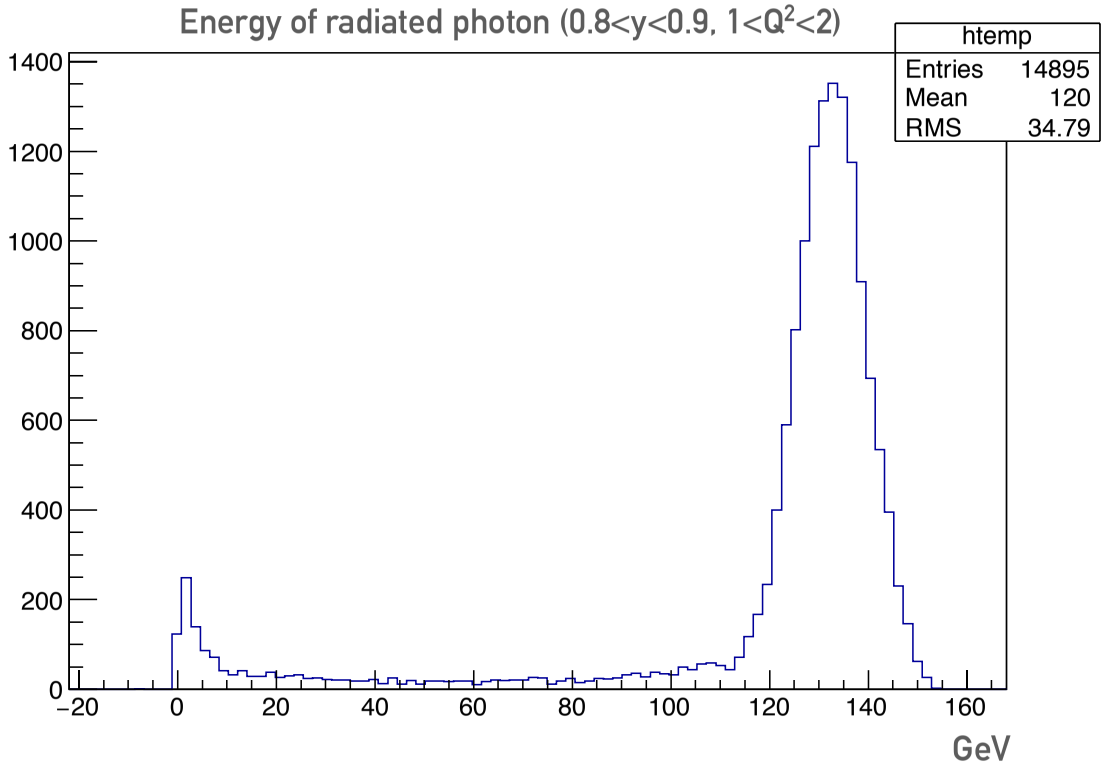
- One for the incident lepton (s-peak)
- One for the outgoing lepton (p-peak)

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.



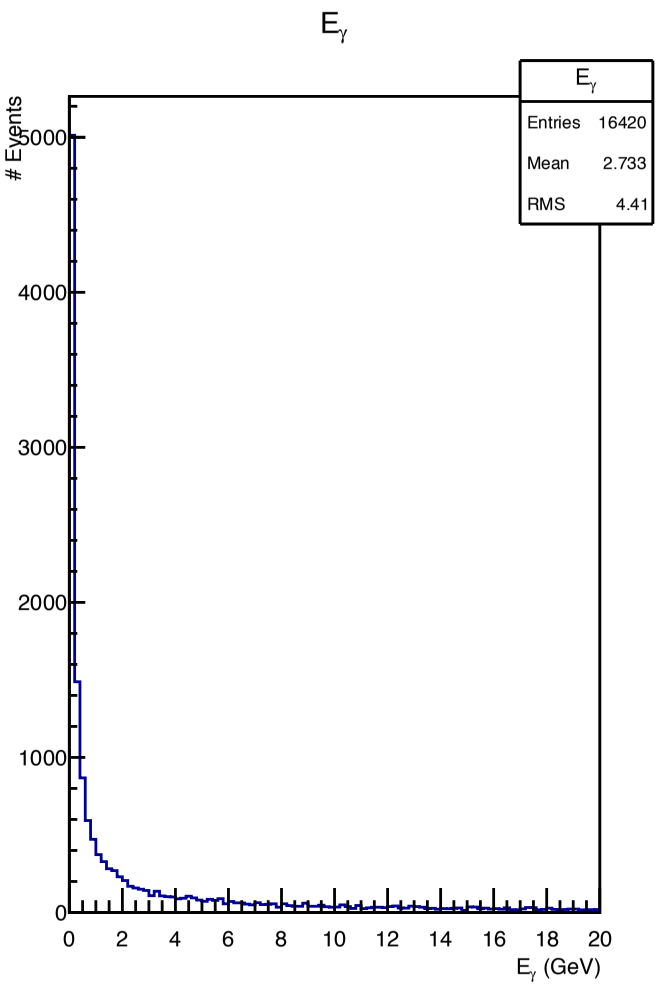
Can we use another MC generator for radiative events and see if it reproduces the results of RADGEN ?

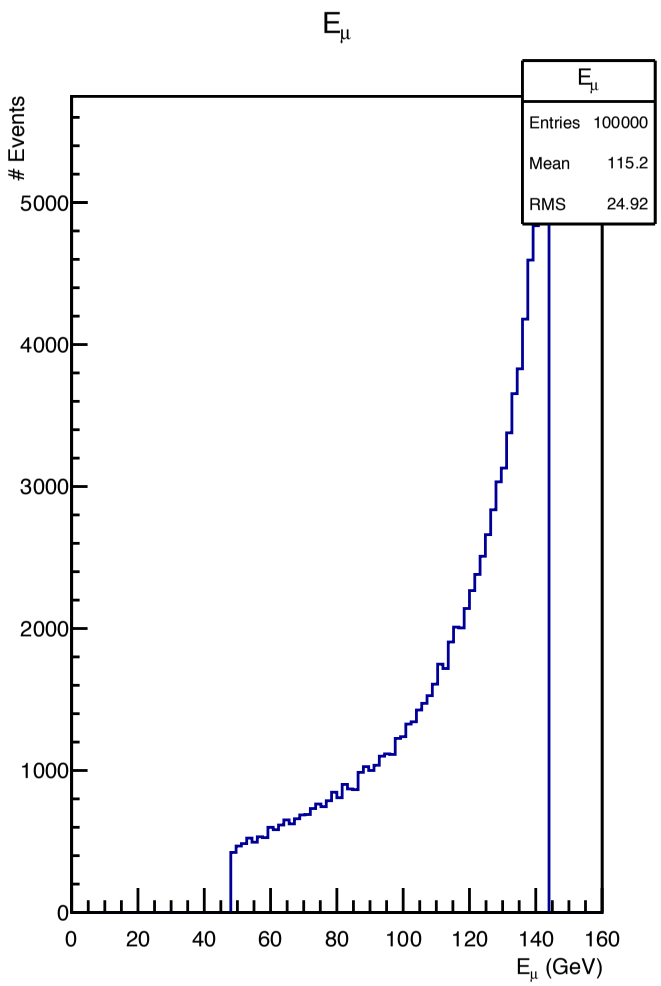
DJANGOH

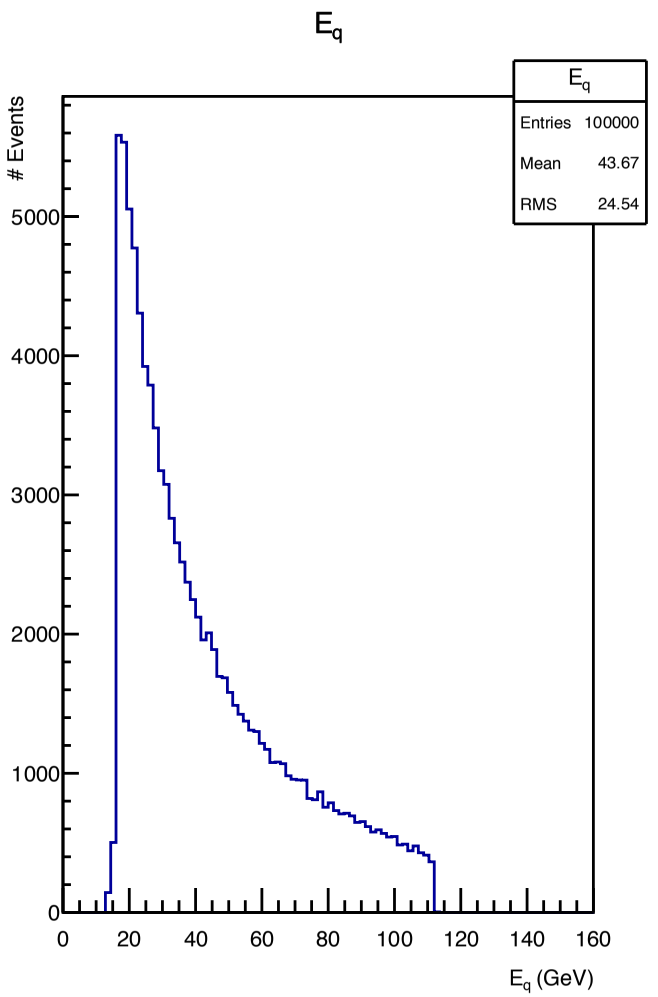
DJANGOH, concatenation of DJANGO and HERACLES :

- 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.

SELF-CONSISTENCY OF DJANGOH

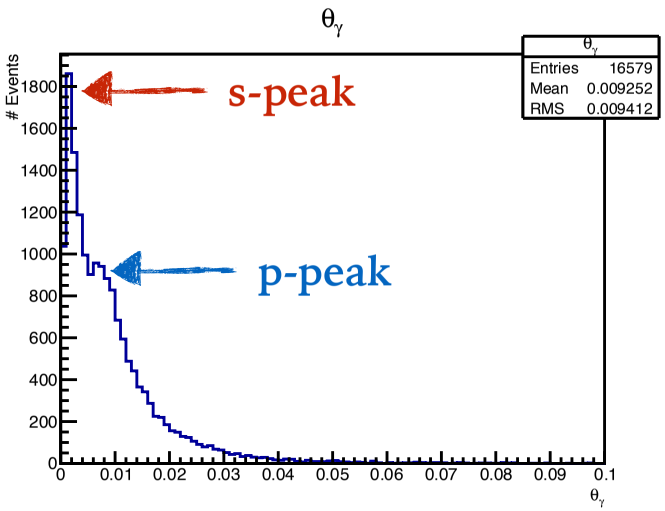


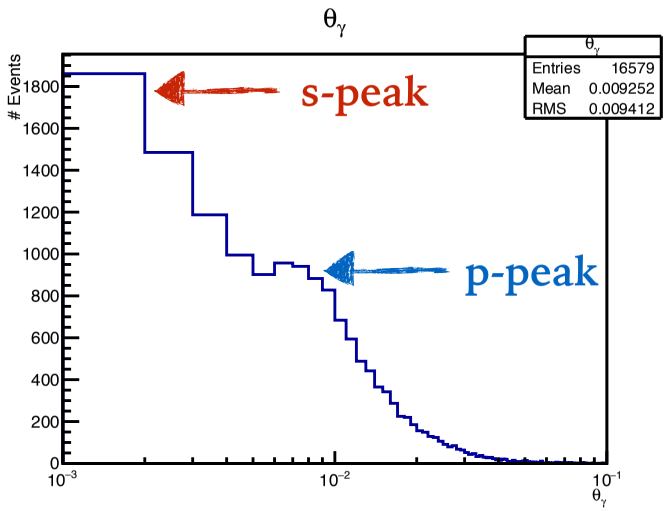


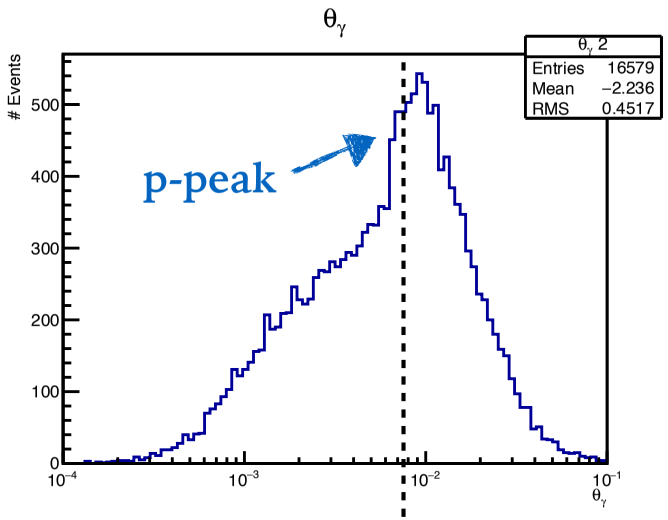


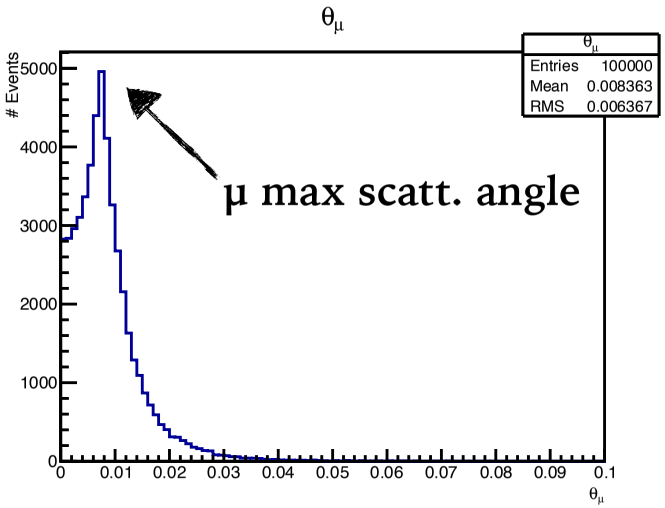
Energy distribution for radiated photon, outgoing muon and struck quark.

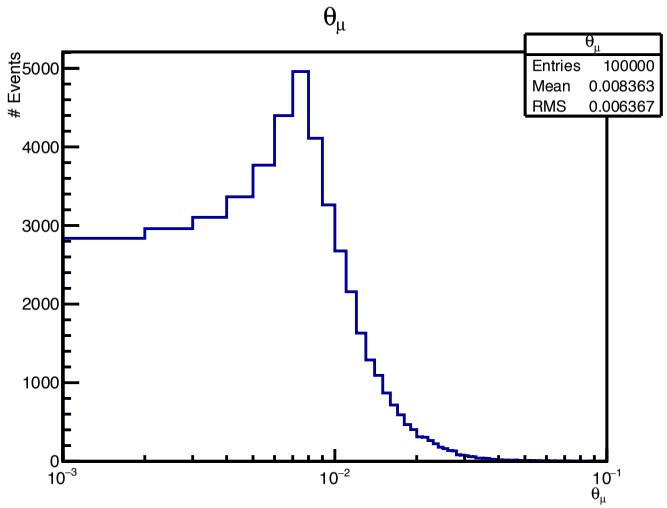
SELF-CONSISTENCY OF DJANGOH

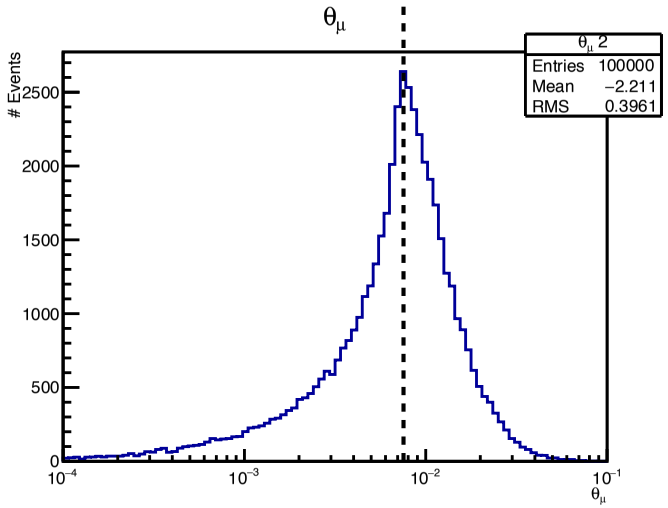








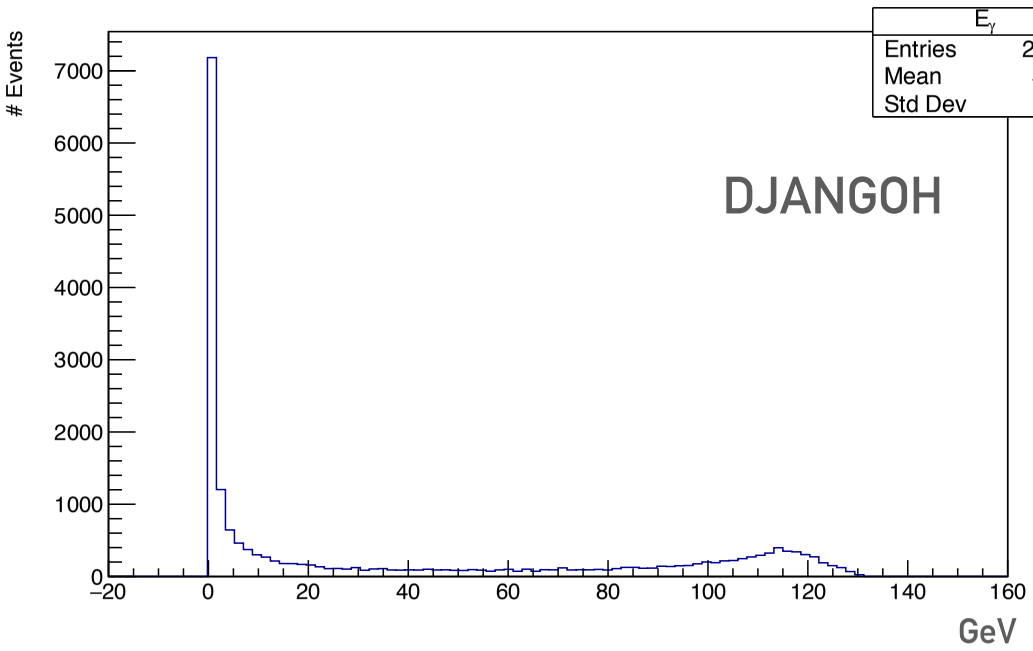


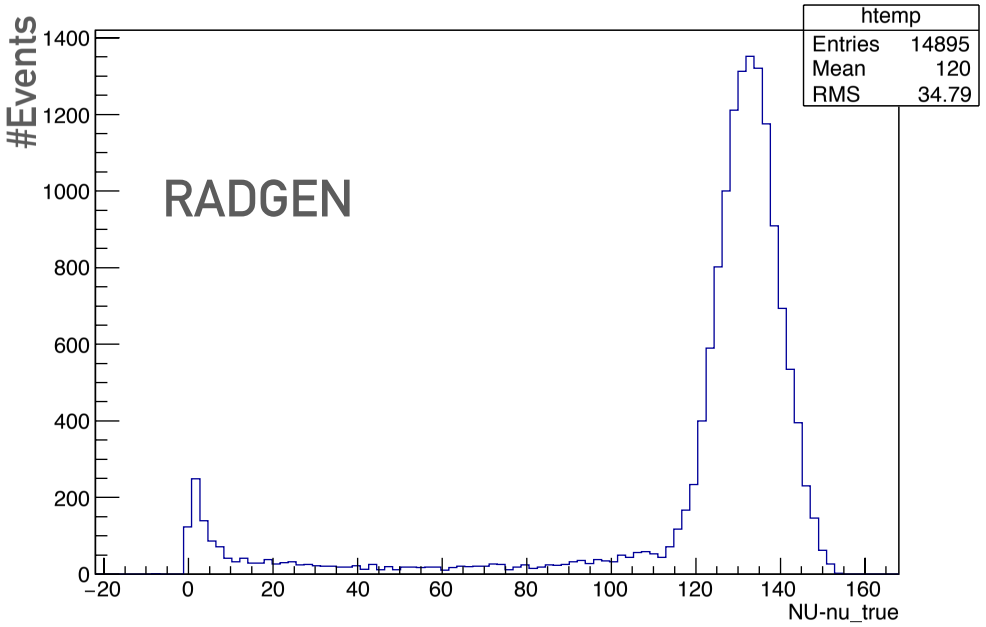


Comparison between θ_γ and θ_μ : p-peak and muon peak scattering angle should match

DJANGOH/RADGEN COMPARISON

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

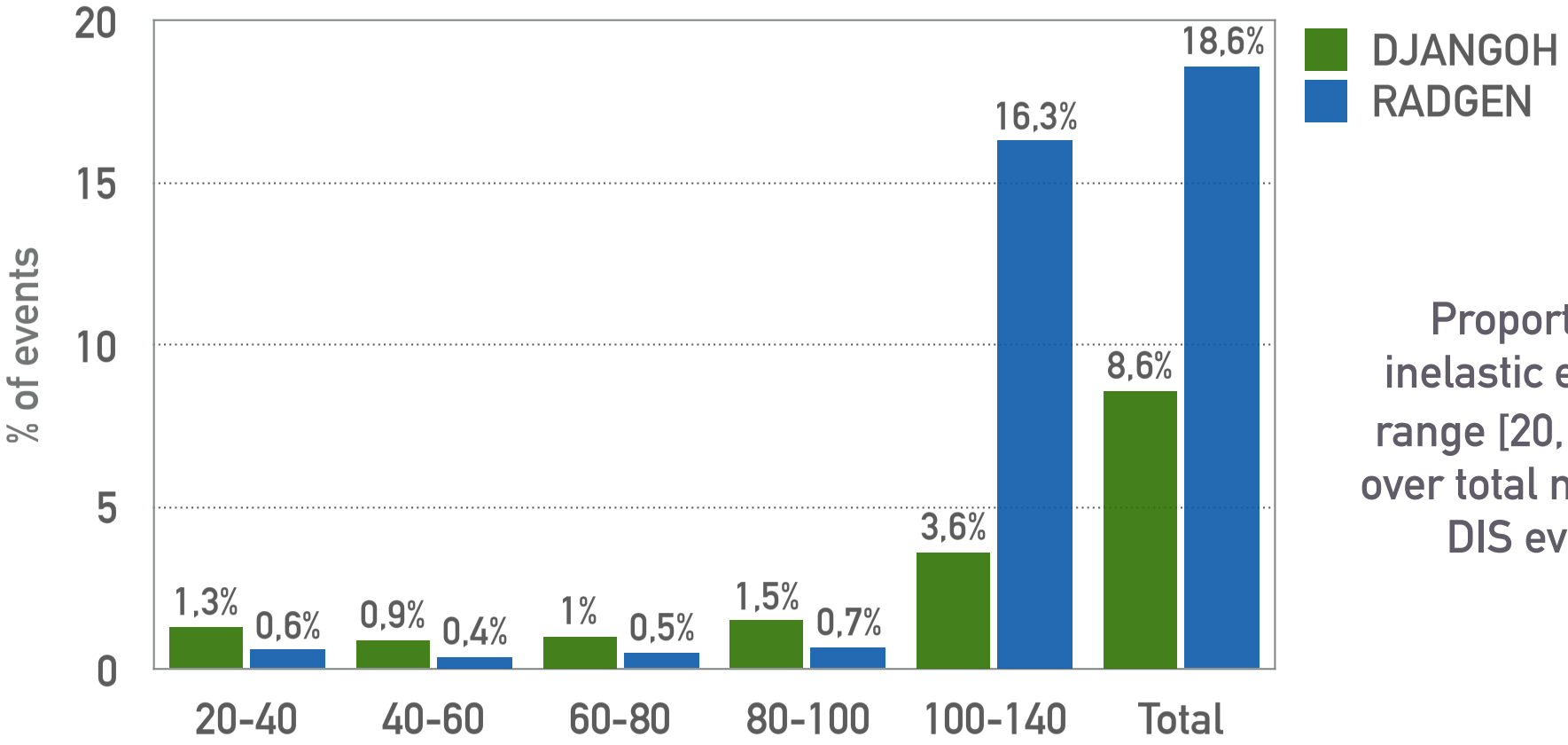




First observation :

- 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.

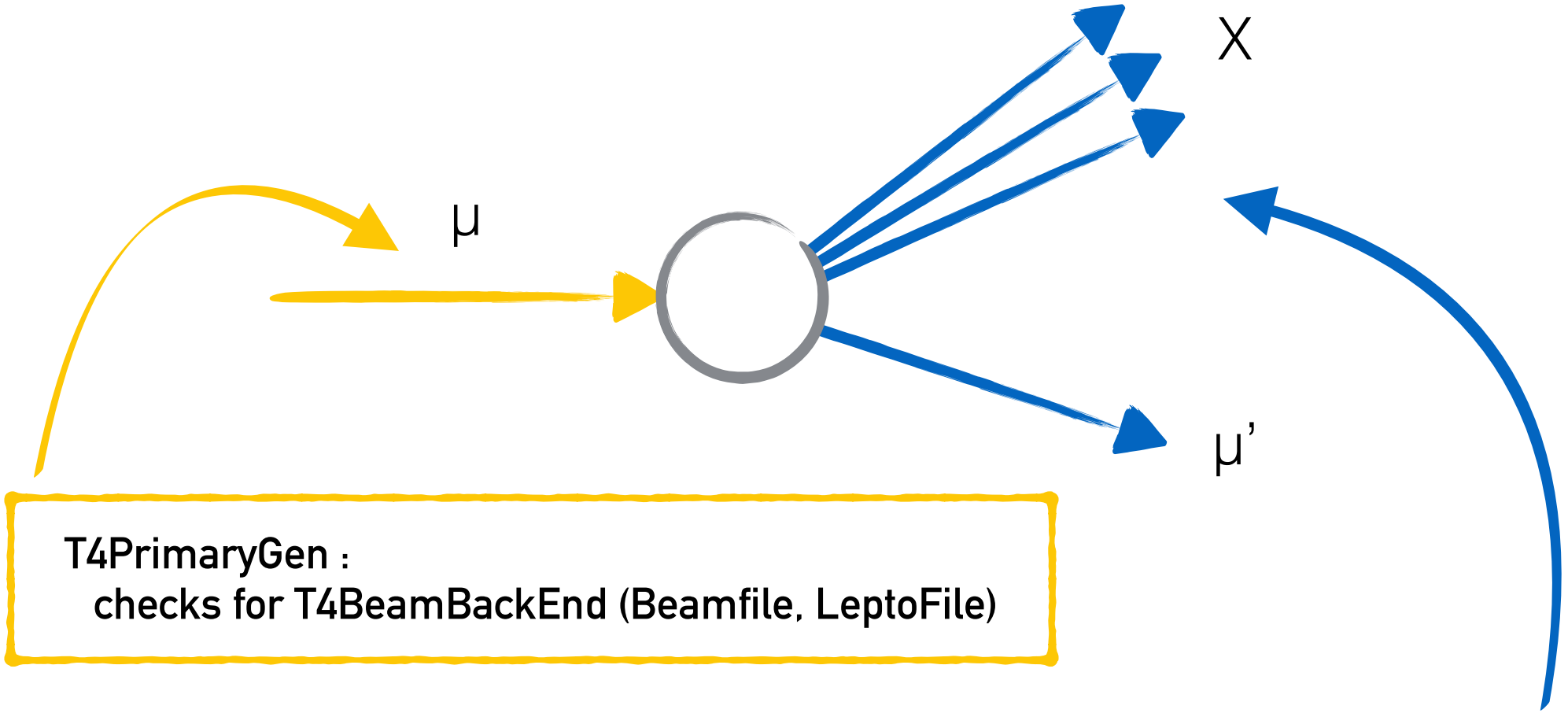
DJANGOH produces less hard photons than RADGEN

TGEANT

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

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

INCOMING BEAM PARTICLE IN TGEANT



T4PrimaryGen :
checks for T4BeamBackEnd (Beamfile, LeptoFile)

T4ProcessBackEnd :
generates process with selected generator and
creates outgoing particles recovered from generator

WAYS TO IMPLEMENT A GENERATOR IN TGEANT

'Internal' generator

Pythia
HepGen++

- C++ Interface
- Perfect conservation of P(p,E)
- Need beamfile for primary generator in TGEANT

OR

'External' generator

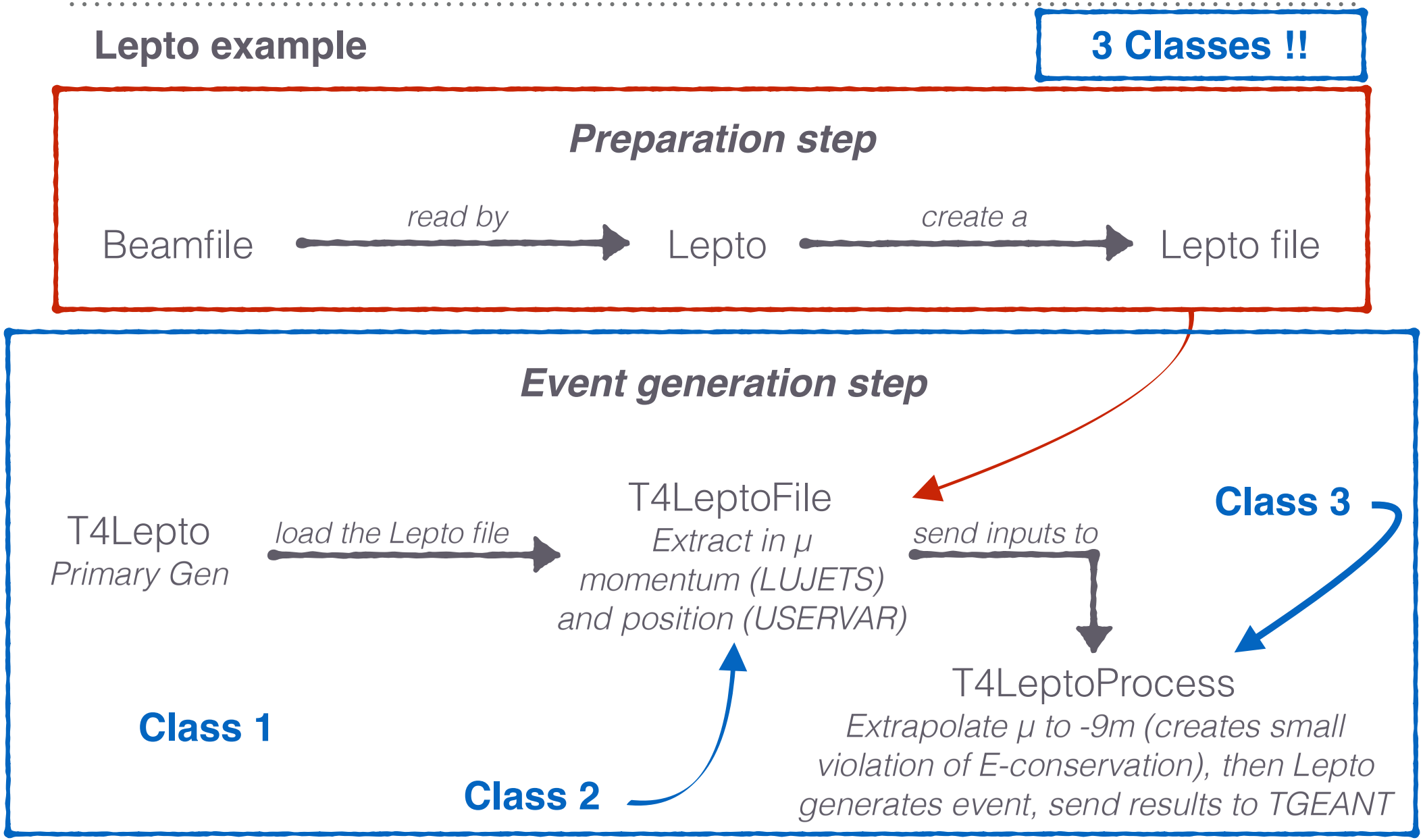
Lepto

- Beamfile read by standalone generator
- Primary generation and process infos in one file
- P(p,E) not perfectly conserved (due to extrapolation to -9m..)

EXTERNAL IMPLEMENTATION AND ITS PROBLEMS

Lepto example

3 Classes !!



EXTERNAL IMPLEMENTATION AND ITS PROBLEMS

Applicated to Djangoh :

- At least 3 classes to implement
- Create a new file format to create the equivalent of the Lepto file for Djangoh
- Modify Djangoh so we can do backward propagation of the incoming μ
- Necessity to recover the output of LUJETS via a file
- Lot of access to file = Lot of computing time wasted

Conclusion : External implementation isn't fitted to Djangoh !

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

Number of classes needed : only 2 !

Interface Class : TDjangoh

- Is a standalone class (depends a bit from ROOT)
- 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

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

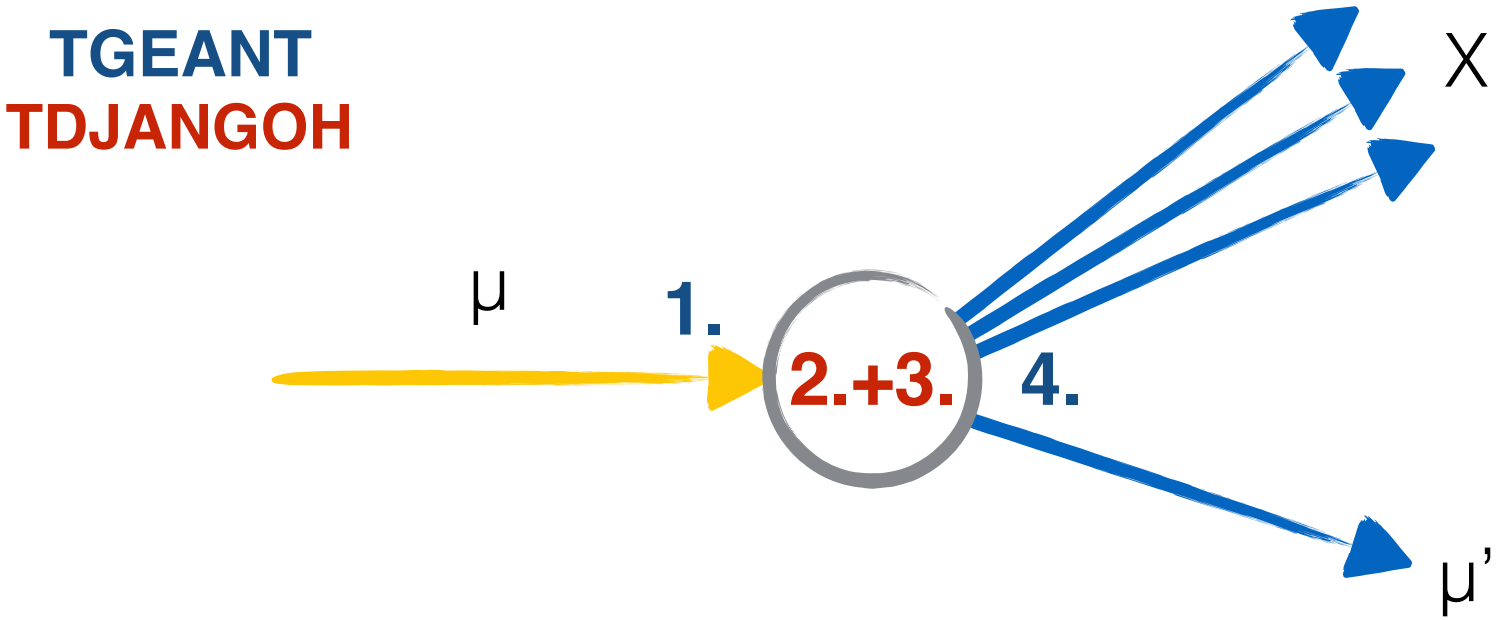
```
Event Generation..
 RANLUX LUXURY LEVEL SET BY RLUXGO : 4      P= 389
 RANLUX INITIALIZED BY RLUXGO FROM SEEDS      2244          0          0
Event Generated !

Total number of particles : 16
Xbj : 0.0427037 y : 0.281150 Q2 : 3.60494
```

Results from Fortran subroutine are well passed to the Interface

PROCESS CLASS

Typical event generation procedure

1. Configure TDjangoh with infos from beamfile.
2. Generate event with TDjangoh.
3. Recover Lujets.
4. Kills all incoming particles and creates new particles according to the content of Lujets.



OPTIMIZING TDJANGOH

Djangoh was designed :

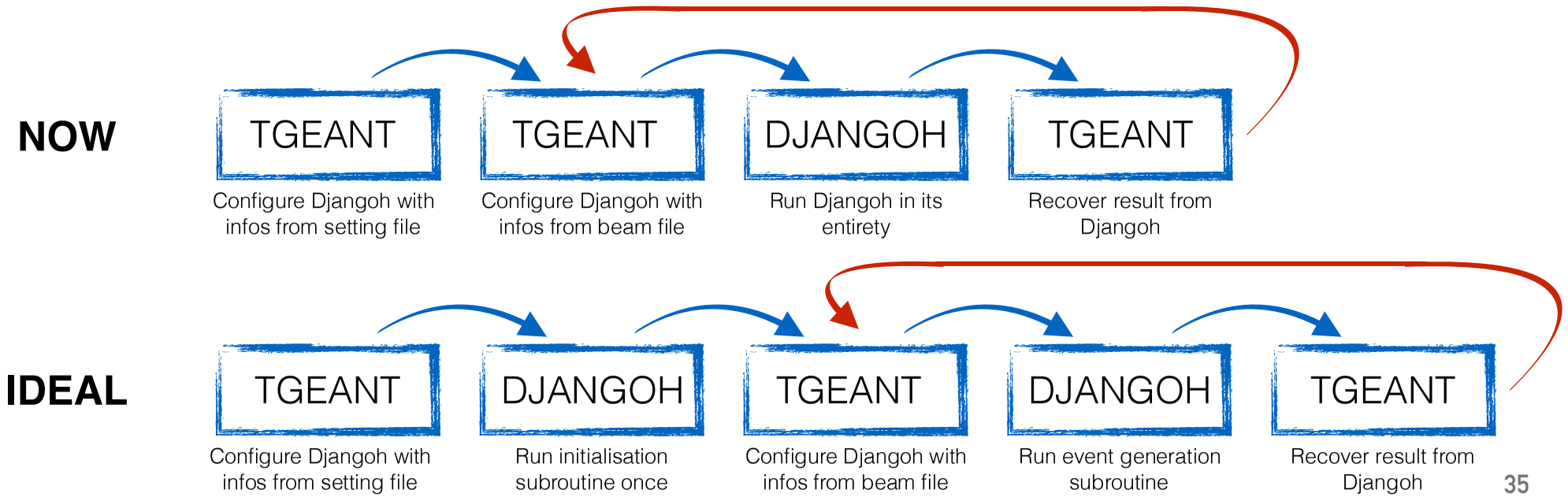
- to work with only one input of incoming lepton energy
- to then run multiple event based on this unique energy

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

MODIFICATION OF DJANGOH

SUBROUTINE HSINIT()

Receive input from Interface

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

- Overall : separate into subroutines and make sure subroutines are sharing correctly their infos
- HSINIT 1st part : Nothing special to do
- HSINIT 2nd part : Looping over integration for each step of the grid

SUBROUTINE HSEVTG()

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

- HSEVTG : Only modification is to pick the right entry in the grid according to input energy.

SUBROUTINE HSRCAP()

Do a sum-up of all generated events

- HSRCAP : Do a recap of all generated events (% of RE, quark composition, etc.)

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

3    USE ISO_C_BINDING

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

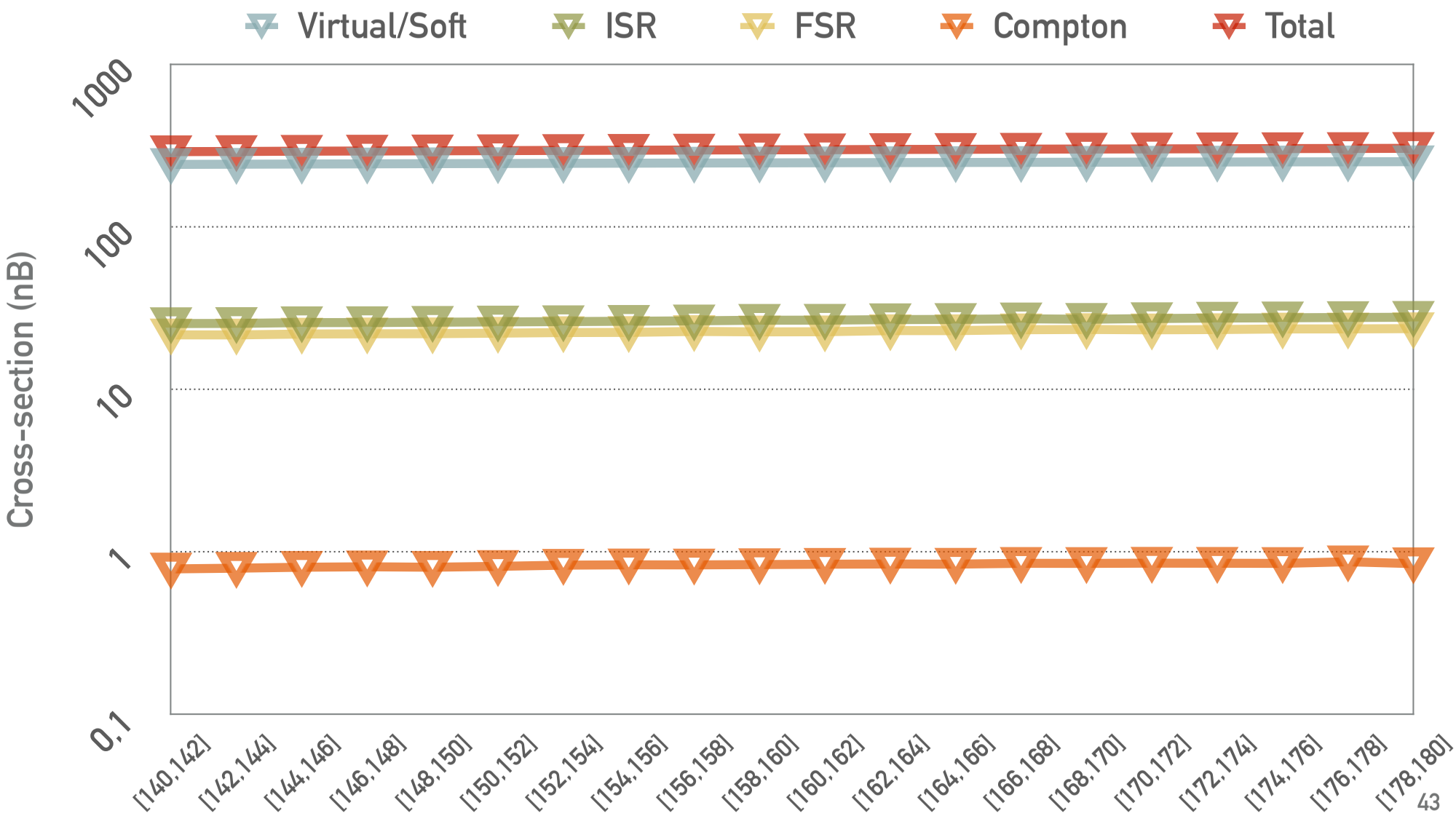
```
1157  C*********************************************************************
1158  C               INTEGRATION / INITIALIZATION FOR EVENT SAMPLING
1159  C               DETERMINATION OF GLOBAL/LOCAL MAXIMA
1160  C*********************************************************************
1161
1162        GDSIZE=4
1163        GDSDDV=4.0
1164        GDSCLE=2.0
1165        GDMEAN=160.0
1166  C---LOOPY-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,EPSO,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       &          EPSO
```

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*



EVENT GENERATION TIME IMPROVEMENT

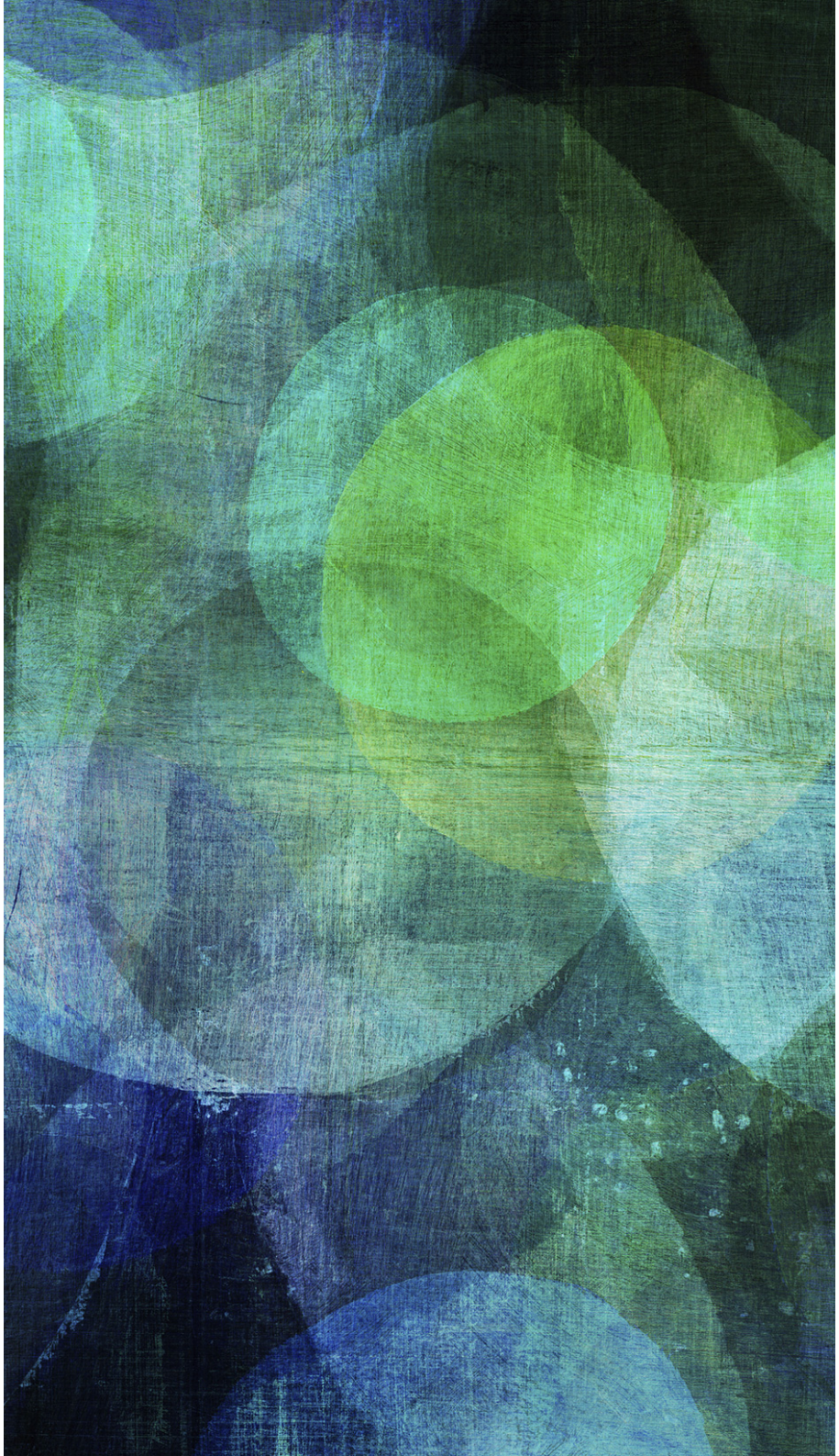
TDjangoh @ event generation :

10s before 0.1s after (fact.100)

TGEANTxTDjangoh @ event generation :

10-20s before 1-3s after (fact. 10)

Generation time improvement
now limited by TGEANT



CONCLUSION

- Renormalization and Radiative Corrections are often seen as complicated and black magic matters, but allow us to have precise results that sticks to the physics one wants to describe.

- One can always recover the true cross-section from the one observed thanks to mathematical objects like radiators.

- Thanks to these mathematical descriptions, possible to create event generators like Djangoh taking care of radiative events, allowing afterwards to calculate radiative corrections to apply to physical results.

- As C/C++ is modular and versatile, possible -not without work !- to port Djangoh as a event generator for TGEANT.