

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

🛡 Security

📊 Insights

⚙ Settings

A C++ Interface to DJANGO

Manage topics

🕒 363 commits

🔗 2 branches

🏷 0 releases

👤 1 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Create or edit file

🐱 nipierre Hotfix for pt correction computation

Test commit at 14:12 on 2019-07-17

📁 TGLA IT script Added TGLA IT script to retrieve DJANGO version

📁 doc Updated doxygen documentation.

📁 include Fix.

📁 src Fix.

📁 test Fix.

📁 utils Hotfix for pt correction computation

📄 DOXYFILE.md Documented

📄 Makefile Test.

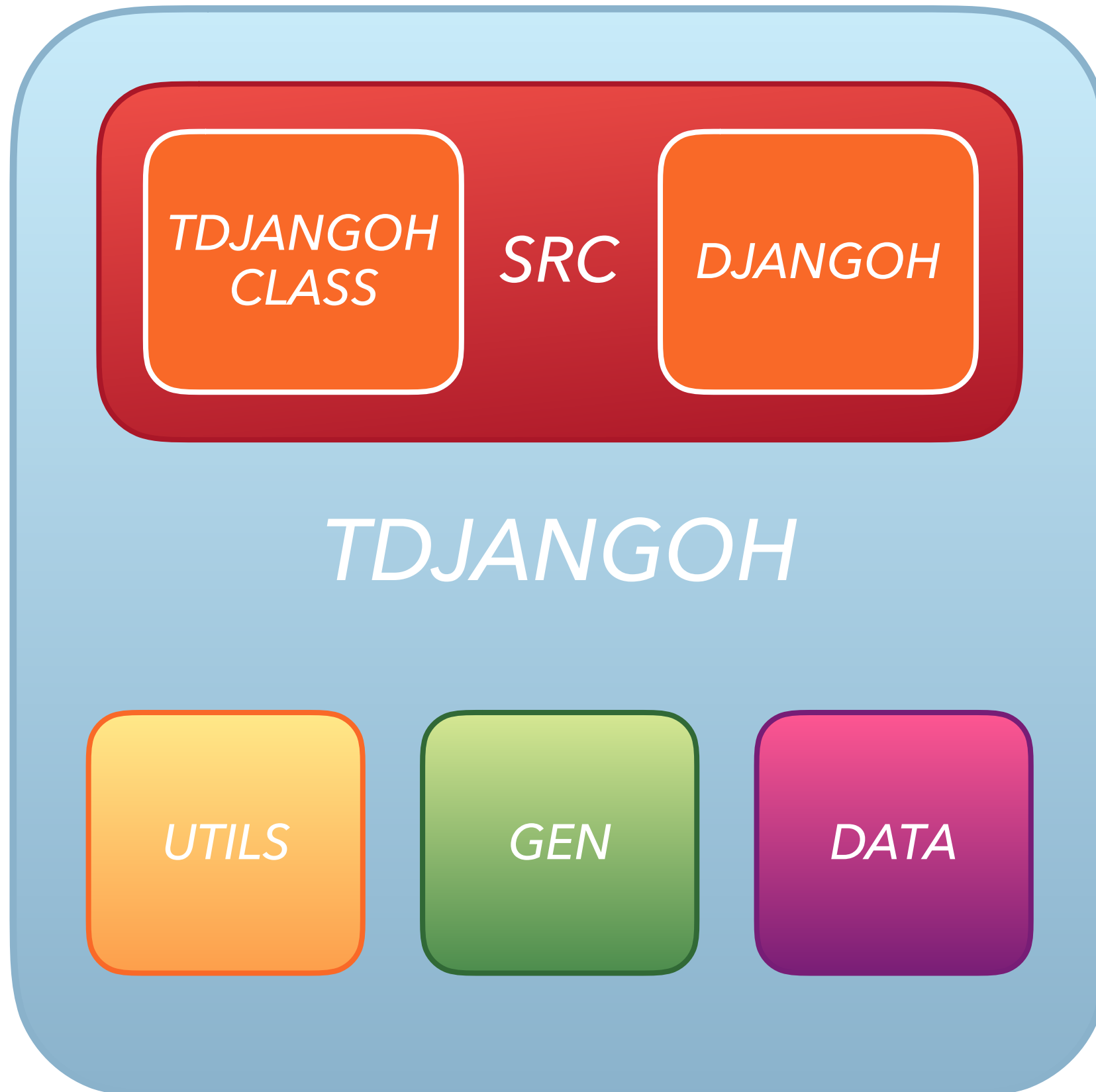
📄 PATCHNOTE.md New Patchnote

📄 README.md New Patchnote

TDJANGO : A C++ MONTE-CARLO EVENT GENERATOR WITH RADIATIVE EVENTS

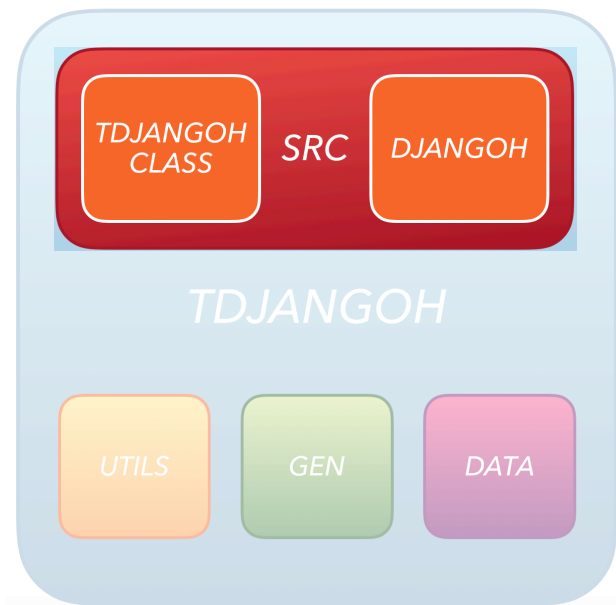
NICOLAS PIERRE - WORKSHOP

JULY 17, 2019



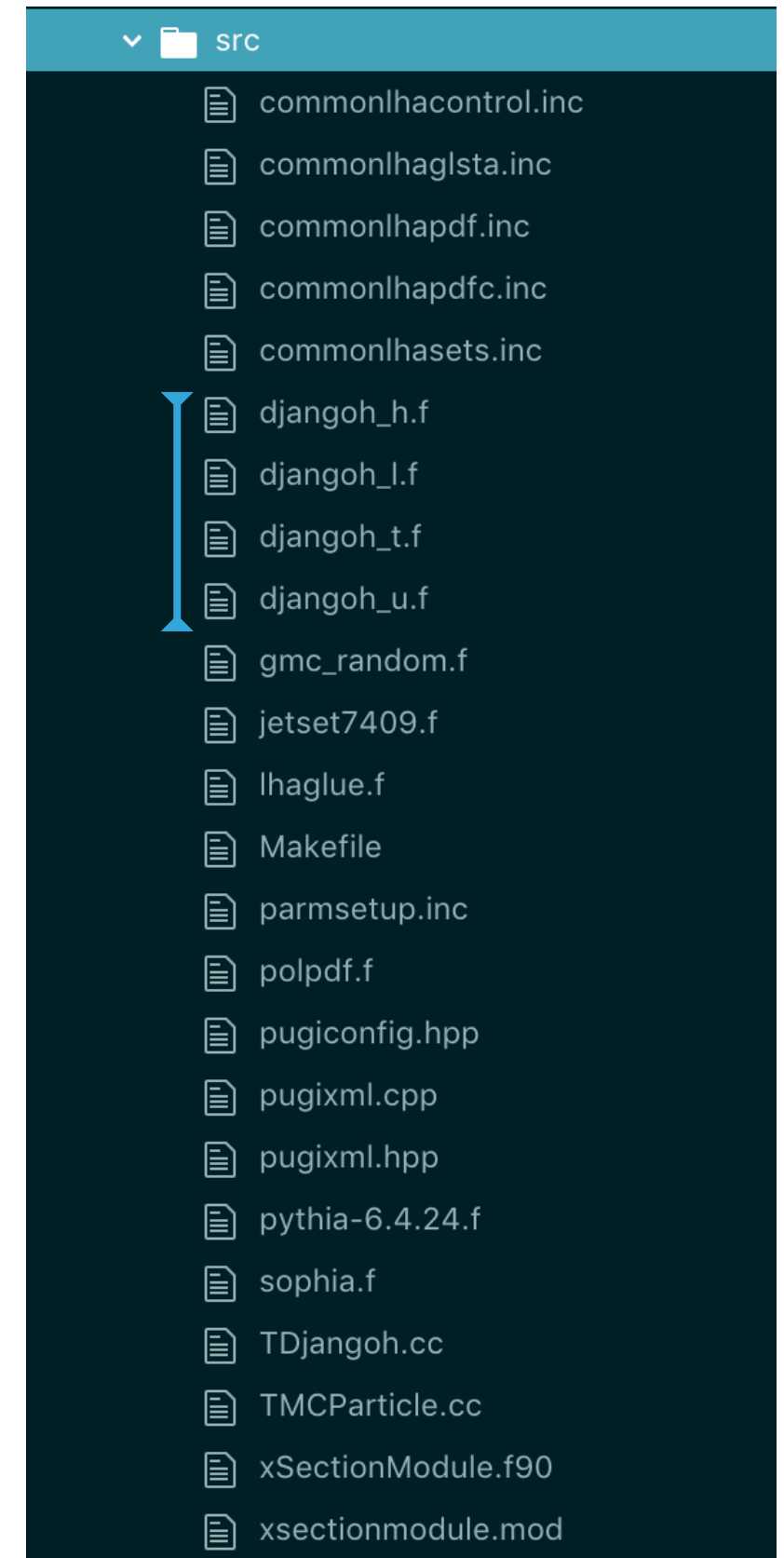
TDJANGO divided in 4 parts :

- ❖ **SRC**: contains source code of the generator and the C++ wrapper.
- ❖ **GEN**: contains applets to run event generation.
- ❖ **UTILS**: contains analysis tools and plotting devices.
- ❖ **DATA**: contains input file for TDJANGO and RC files from TERAD.



- ❖ **SRC**: contains source code of the generator and the C++ wrapper.
- ❖ **DJANGO** : FORTRAN code. Should be kept as a 'black box'.

- ❖ **djangoh_h.f**: Heracles file. Subroutine HSINPT() called by TDJANGO
- ❖ **djangoh_l.f**: DJANGO6 file. LEPTO65 + DJANGO routines.
- ❖ **djangoh_u.f**: User analysis routines. Program performance.
- ❖ **djangoh_t.f**: Legacy, unused.



❖ Non-radiative event : HERACLES generates events at the parton level. Fragmentation and hadronization done by DJANGO6.

❖ Radiative part :

❖ **Initial State Radiation (ISR)** : two-step process $l \rightarrow \tilde{l} + \gamma, \quad \tilde{l} + p \rightarrow l' + \gamma + X$

After emission of the real photon, as in non-radiative event.

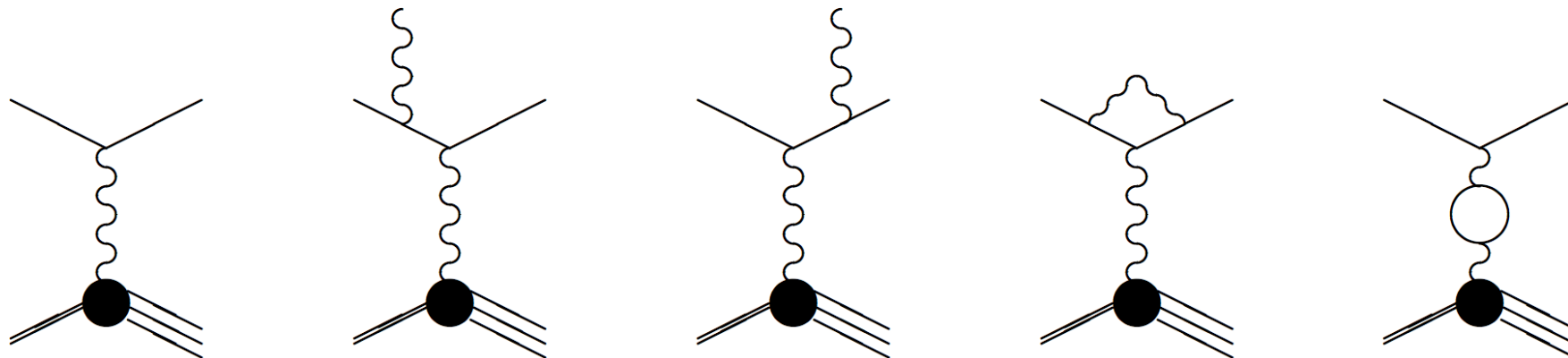
❖ **Final State Radiation (FSR)** : similar than ISR $l + p \rightarrow l' + \gamma + X, \quad l' \rightarrow \tilde{l}' + \gamma$

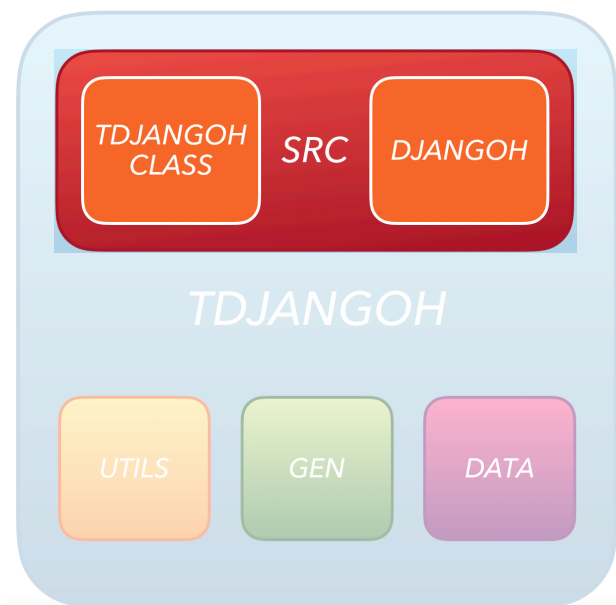
After producing events as in non-radiative case, emission of real photon

❖ **Compton events**

❖ Typically low Q^2 and low W_{had} events. Fraction of a percent of the events for $W_{\text{had}} > 2 \text{ GeV}/c^2$

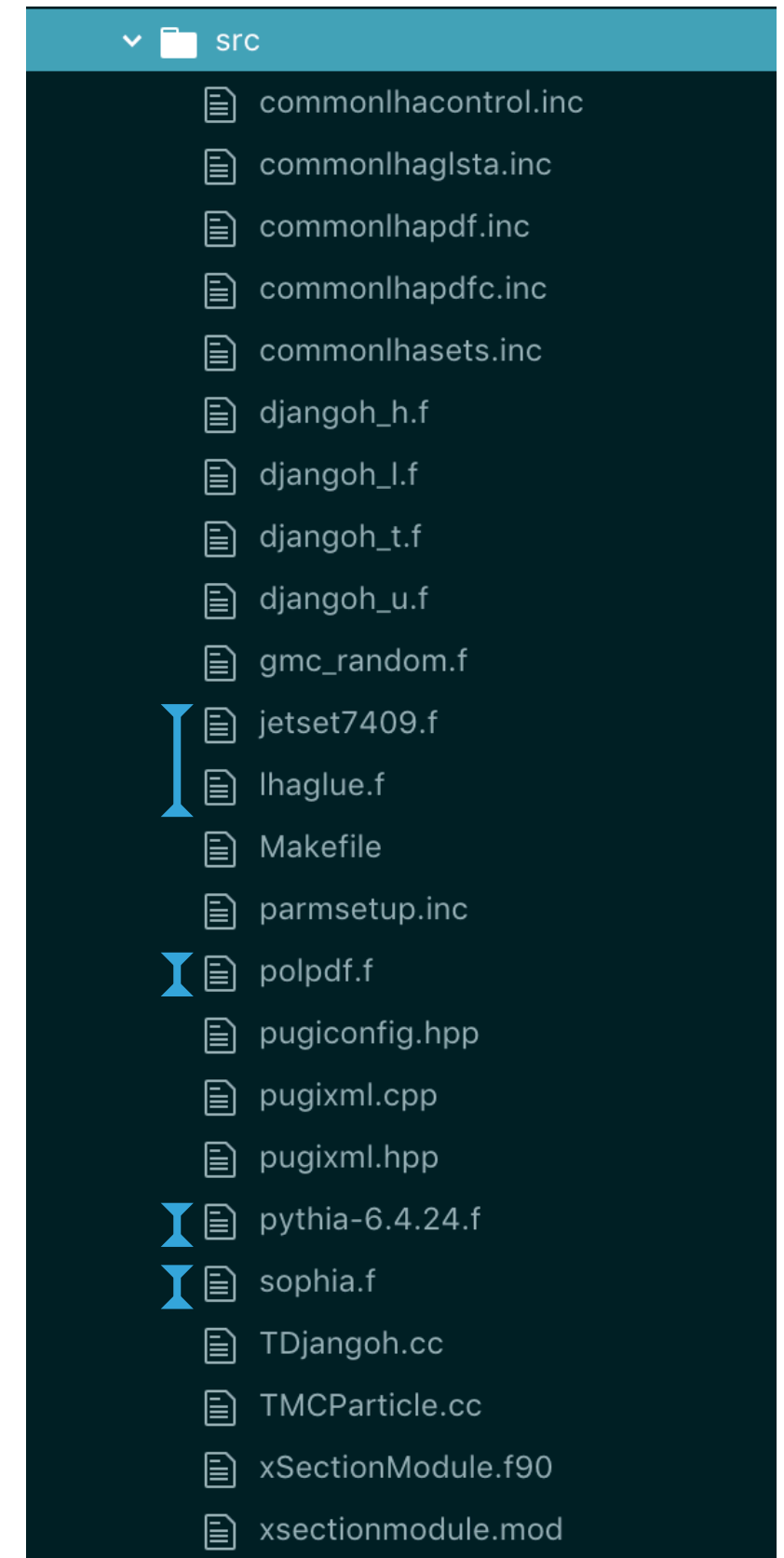
❖ For $W_{\text{had}} < 2 \text{ GeV}/c^2$, generation of final state performed by SOPHIA.

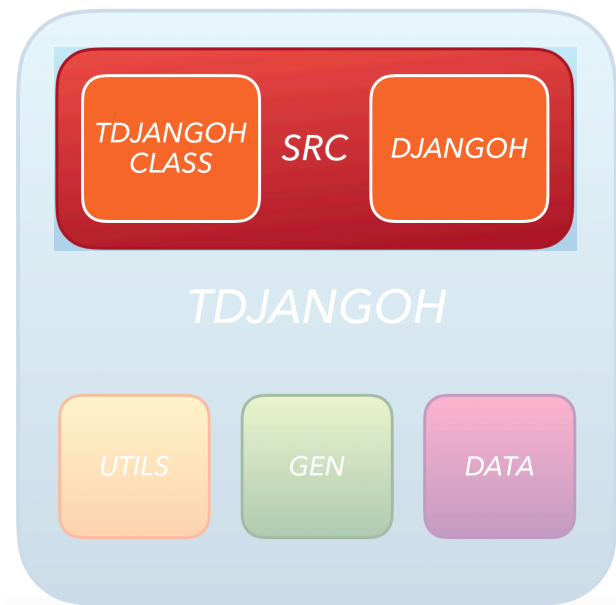




- ❖ **SRC**: contains source code of the generator and the C++ wrapper.
- ❖ **DJANGO** : FORTRAN code. Should be kept as a 'black box'.

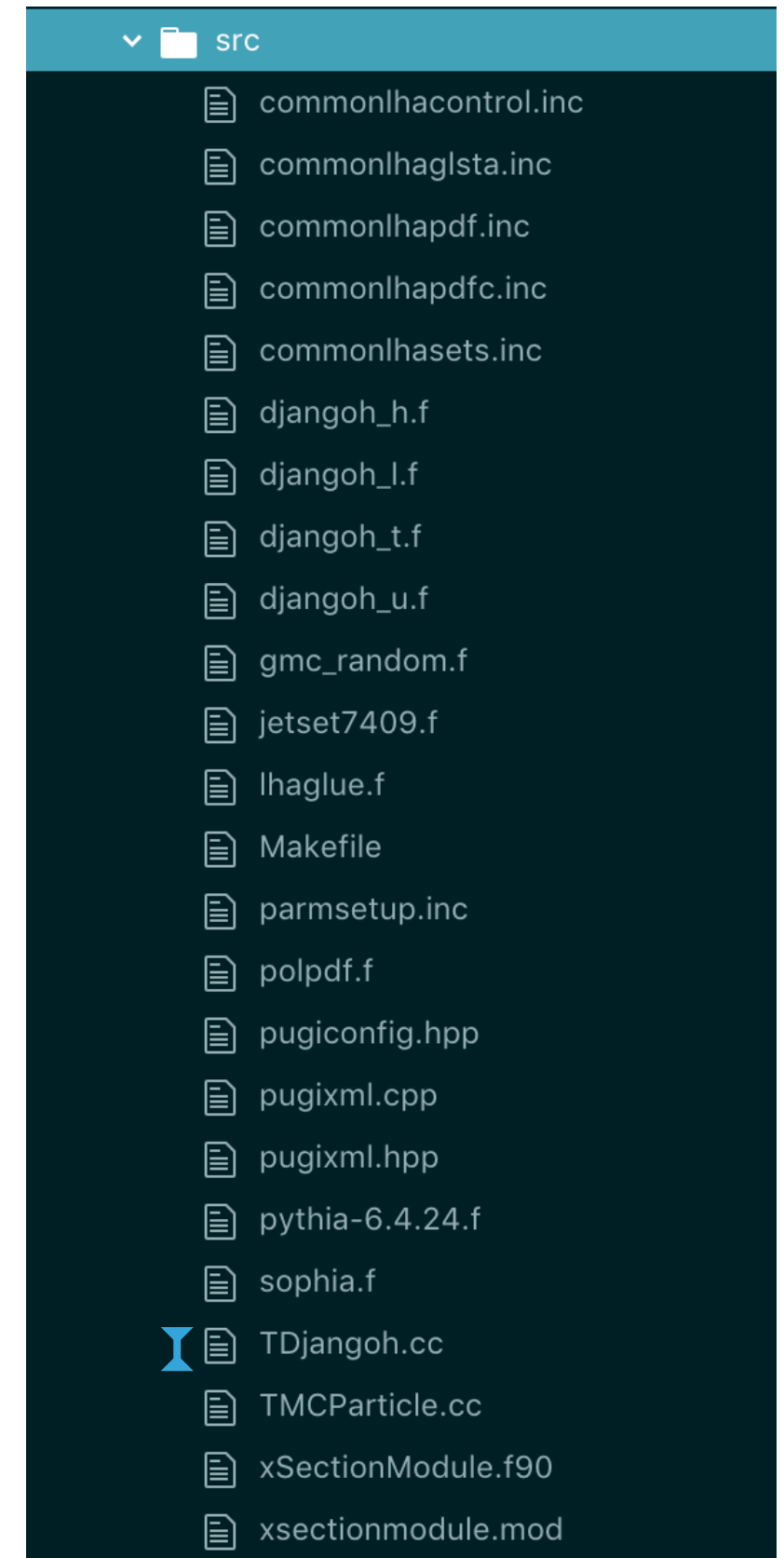
- ❖ **JETSET, PYTHIA, SOPHIA** for hadronic final states.

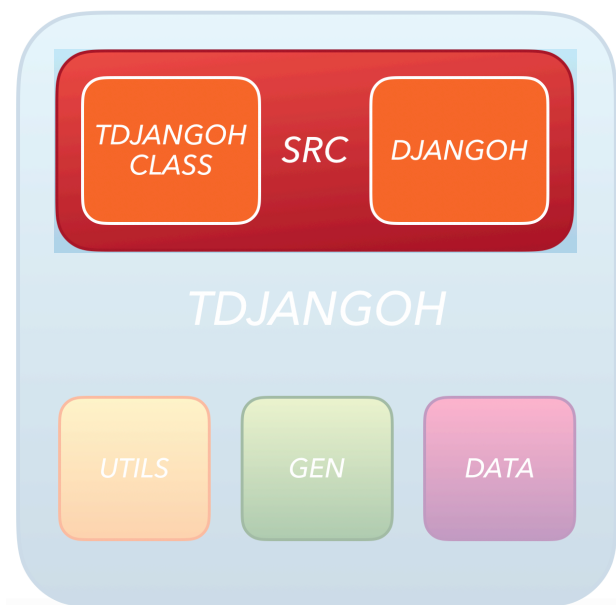




- ❖ **SRC**: contains source code of the generator and the C++ wrapper.

- ❖ **TDJANGO**, C++ interface.
- ❖ For more details on the methods available in TDJANGO class, see doxygen documentation.





Input for TDJANGOH stored in an XML file.

- ❖ **EL-BEAM**, infos about the beam.
- ❖ **KINEM-CUT**, kinematical cuts (on leptonic variables). 'icut' determines the cut applied (for 3 all are applied).
- ❖ **GSW-PARAM**, parameters of the theory (put lparin2=0 to cancel virtual corrections to Born XS)

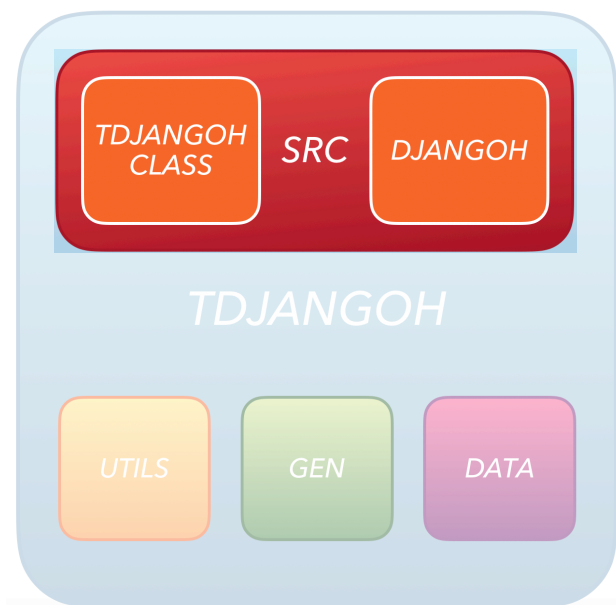
```

<!-- EL-BEAM -->
<Codeword name="EL-BEAM">
  <Data name="polari" value="0.0"/>
  <Data name="beam" value="mu+"/>
</Codeword>

<!-- KINEM-CUT -->
<Codeword name="KINEM-CUTS">
  <Data name="icut" value="3"/>
  <Data name="ixmin" value="0.001"/>
  <Data name="ixmax" value="0.95"/>
  <Data name="iymin" value="0.01"/>
  <Data name="iymax" value="0.95"/>
  <Data name="iq2min" value="0.8"/>
  <Data name="iq2max" value="80"/>
  <Data name="iwmin" value="4"/>
</Codeword>

<!-- GSW-PARAM -->
<Codeword name="GSW-PARAM">
  <Data name="lparin1" value="2"/>
  <Data name="lparin2" value="1"/>
  <Data name="lparin3" value="3"/>
  <Data name="lparin4" value="1"/>
  <Data name="lparin5" value="0"/>
  <Data name="lparin6" value="0"/>
  <Data name="lparin7" value="2"/>
  <Data name="lparin8" value="1"/>
  <Data name="lparin9" value="1"/>
  <Data name="lparin10" value="1"/>
  <Data name="lparin11" value="1"/>
</Codeword>

```

- ❖ **GD-OPT**, XS grid options
(Mean value in GeV, spread in GeV, number of bin)
- ❖ **EGAM-MIN**,
- ❖ **INT-OPT-NC**, number of integration for channels contributing to the XS.
(Born+Virtual, ISR, FSR, Compton, quark line, el, el ISR, el FSR, el Compton)

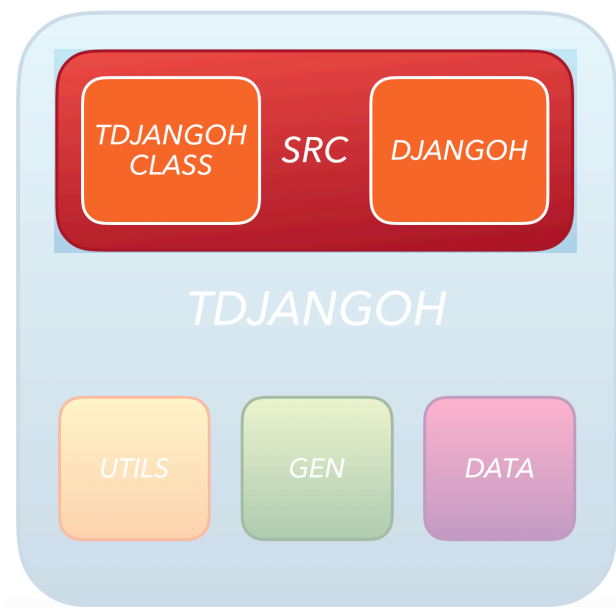
- ❖ **SAM-OPT-NC**, activation of the channels.

```
<!-- SAM_OPT_NC -->
<Codeword name="SAM-OPT-NC">
  <Data name="isnc2" value="1"/>
  <Data name="isnc31" value="1"/>
  <Data name="isnc32" value="1"/>
  <Data name="isnc33" value="1"/>
  <Data name="isnc34" value="0"/>
  <Data name="isel2" value="1"/>
  <Data name="isel31" value="1"/>
  <Data name="isel32" value="1"/>
  <Data name="isel33" value="1"/>
</Codeword>
```

```
<!-- GD-OPT -->
<Codeword name="GD-OPT">
  <Data name="gdmean" value="160.0"/>
  <Data name="gdsddv" value="20"/>
  <Data name="gdsdsize" value="20"/>
</Codeword>

<!-- EGAM-MIN -->
<Codeword name="EGAM-MIN">
  <Data name="egam" value="0"/>
</Codeword>

<!-- INT-OPT-NC -->
<Codeword name="INT-OPT-NC">
  <Data name="inc2" value="1"/>
  <Data name="inc31" value="18"/>
  <Data name="inc32" value="18"/>
  <Data name="inc33" value="18"/>
  <Data name="inc34" value="0"/>
  <Data name="iel2" value="1"/>
  <Data name="iel31" value="18"/>
  <Data name="iel32" value="18"/>
  <Data name="iel33" value="18"/>
</Codeword>
```

- ❖ **NUCLEUS**, infos on the target (energy if beam, polarization, A, Z).
- ❖ **STRUCTFUNC**, choice of structure function (see COMPASS note for more details).
- ❖ **FLONG**, parametrization for F_L
- ❖ **ALFAS**, parametrization for

```

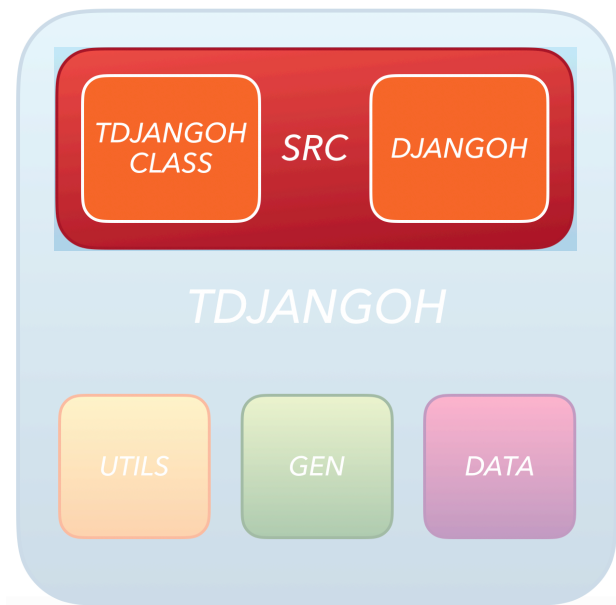
<!-- NUCLEUS -->
<Codeword name="NUCLEUS">
  <Data name="epro" value="0"/>
  <Data name="hpolar" value="0"/>
  <Data name="hna" value="1"/>
  <Data name="hnz" value="1"/>
</Codeword>

<!-- STRUCTFUNC -->
<Codeword name="STRUCTFUNC">
  <Data name="ilqmod" value="0"/>
  <Data name="ilib" value="2"/>
  <Data name="icode" value="21000"/>
</Codeword>

<!-- FLONG -->
<Codeword name="FLONG">
  <Data name="iflopt" value="122"/>
  <Data name="parl11" value="0.01"/>
  <Data name="parl19" value="0.03"/>
</Codeword>

<!-- ALFAS -->
<Codeword name="ALFAS">
  <Data name="mst111" value="1"/>
  <Data name="mst115" value="1"/>
  <Data name="par111" value="0.2"/>
  <Data name="par112" value="0.235"/>
</Codeword>

```



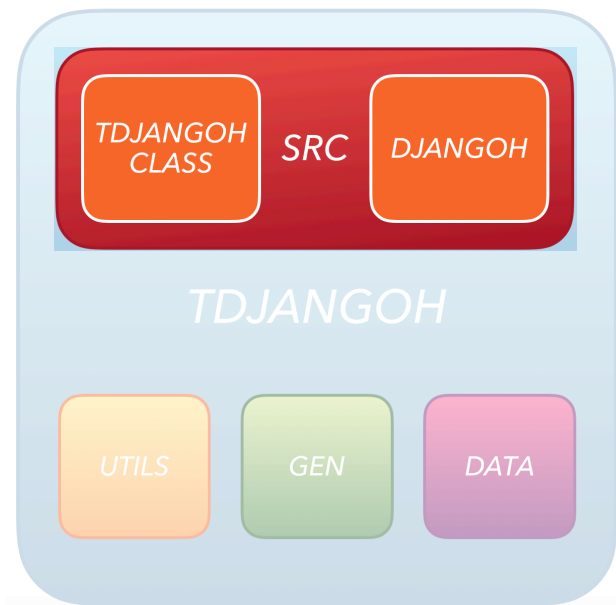
- ❖ **HIGH-PT**, for high p_T tuning.
- ❖ **NFLAVORS**, number of quark flavours.
- ❖ **SOPHIA**, floor on W triggering the use of SOPHIA.

```
<!-- HIGH-PT -->
<Codeword name="HIGH-PT">
  <Data name="parj21" value="0.34"/>
  <Data name="parj23" value="0.04"/>
  <Data name="parj24" value="2.8"/>
  <Data name="parj41" value="0.025"/>
  <Data name="parj42" value="0.075"/>
</Codeword>

<!-- NFLAVORS -->
<Codeword name="NFLAVORS">
  <Data name="npymin" value="1"/>
  <Data name="npymax" value="6"/>
</Codeword>

<!-- RNDM-SEEDS -->
<Codeword name="RNDM-SEEDS">
  <Data name="isdinp" value="-1"/>
  <Data name="isdout" value="-1"/>
</Codeword>

<!-- SOPHIA -->
<Codeword name="SOPHIA">
  <Data name="wsophia" value="1.4"/>
</Codeword>
```



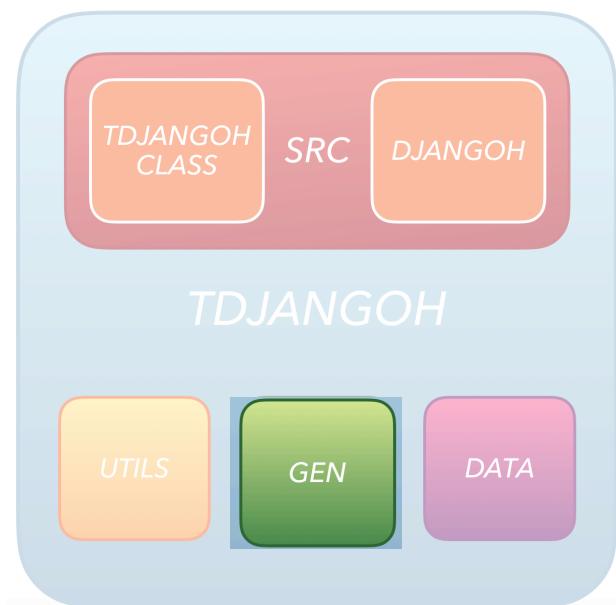
- ❖ **FRAG**, enable fragmentation.
- ❖ **VERBOSE**, enable text printing for generator.
- ❖ **UNFRAG-SAVE**, allow unfragmented states.
- ❖ **FORCE-GEN**, generation of new event if previous fails.

```
<!-- FRAG -->
<Codeword name="FRAG">
  <Data name="lst7" value="1"/>
</Codeword>
```

```
<!-- VERBOSE -->
<Codeword name="VERBOSE">
  <Data name="verboz" value="0"/>
</Codeword>

<!-- UNFRAG-SAVE -->
<Codeword name="UNFRAG-SAVE">
  <Data name="unfrag" value="0"/>
</Codeword>

<!-- FORCE-GEN -->
<Codeword name="FORCE-GEN">
  <Data name="frcgen" value="1"/>
</Codeword>
```



```
cout << FCYN("Instance creation..") << endl;
tDjangoh = new TDjangoh();
cout << FCYN("Instance created !") << endl;
```

Instance creation

```
cout << FCYN("Initialization..") << endl;
tDjangoh->ReadXMLFile(argv[1]);
tDjangoh->Initialize();
cout << FCYN("Initialized !") << endl;
```

Read XML file and initialize

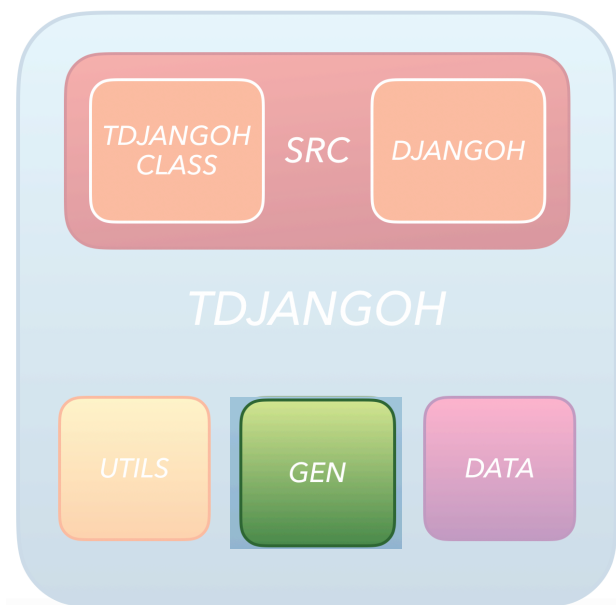
```
cout << FCYN("\n\nEvents Generation (" << NEVENTS << " events)..") << endl;
for(int i=0; i<NEVENTS; i++)
{
```

```
    if(rnd)
        tDjangoh->SetBeam(base_energy+rand()%40,0);
    else
        tDjangoh->SetBeam(base_energy,0);
```

Example of
changing
beam energy

```
tDjangoh->GenerateEvent();
```

Event generation



```

nb = tDjangoh->GetN();
cout << FCYN("Total number of particles : ") << nb << endl;
for(int j=0; j<nb; j++)
{
    cout
    << tDjangoh->GetK(j,1) << " "
    << tDjangoh->GetK(j,2) << " "
    << tDjangoh->GetK(j,3) << " "
    << tDjangoh->GetK(j,4) << " "
    << tDjangoh->GetK(j,5) << " \n"
    << tDjangoh->GetP(j,1) << " "
    << tDjangoh->GetP(j,2) << " "
    << tDjangoh->GetP(j,3) << " "
    << tDjangoh->GetP(j,4) << " "
    << tDjangoh->GetP(j,5) << " \n"
    << tDjangoh->GetV(j,1) << " "
    << tDjangoh->GetV(j,2) << " "
    << tDjangoh->GetV(j,3) << " "
    << tDjangoh->GetV(j,4) << " "
    << tDjangoh->GetV(j,5) << " \n" << endl;
}
x = tDjangoh->GetX();
y = tDjangoh->GetY();
Q2 = tDjangoh->GetQ2();
cout << "Xbj : " << x << " y : " << y << " Q2 : " << Q2 << "\n\n" << endl;

```

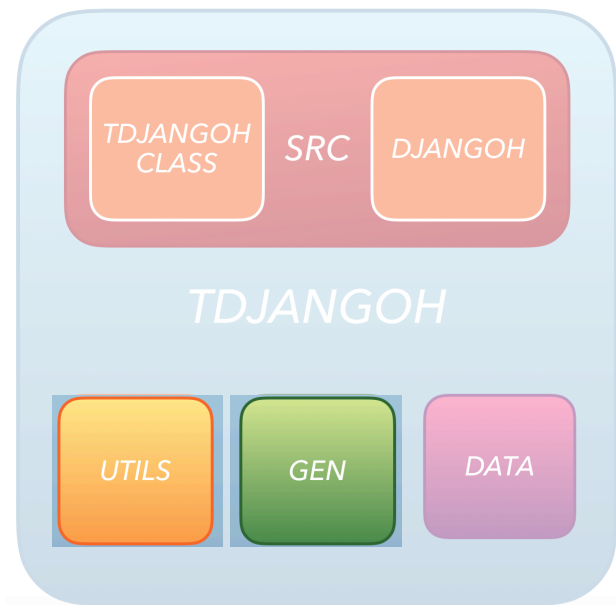
How to access final state particles infos

K: identity

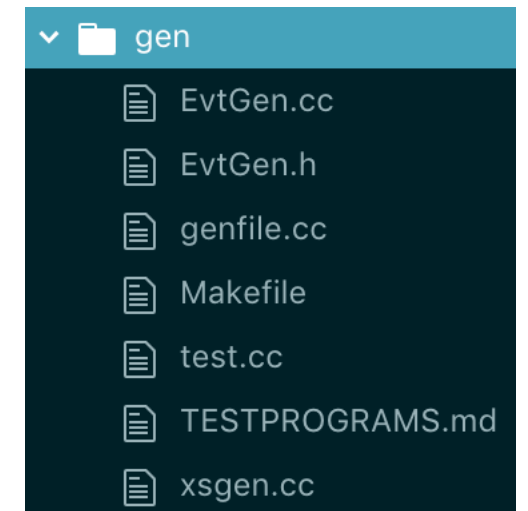
P: momenta

V: vertex position

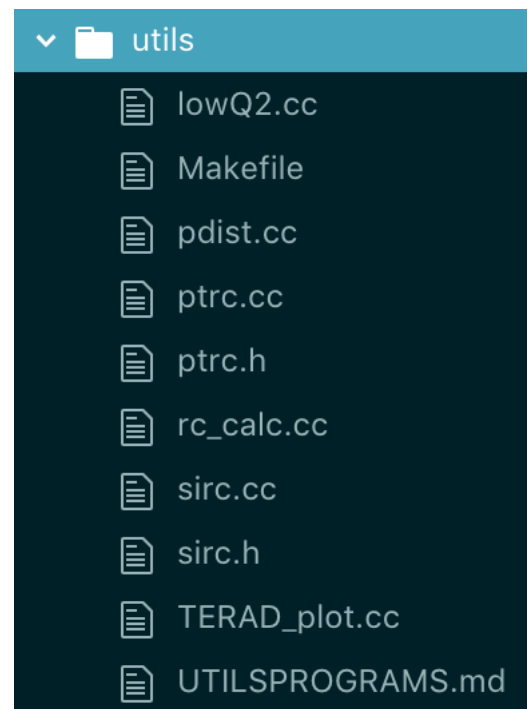
Access events kinematics

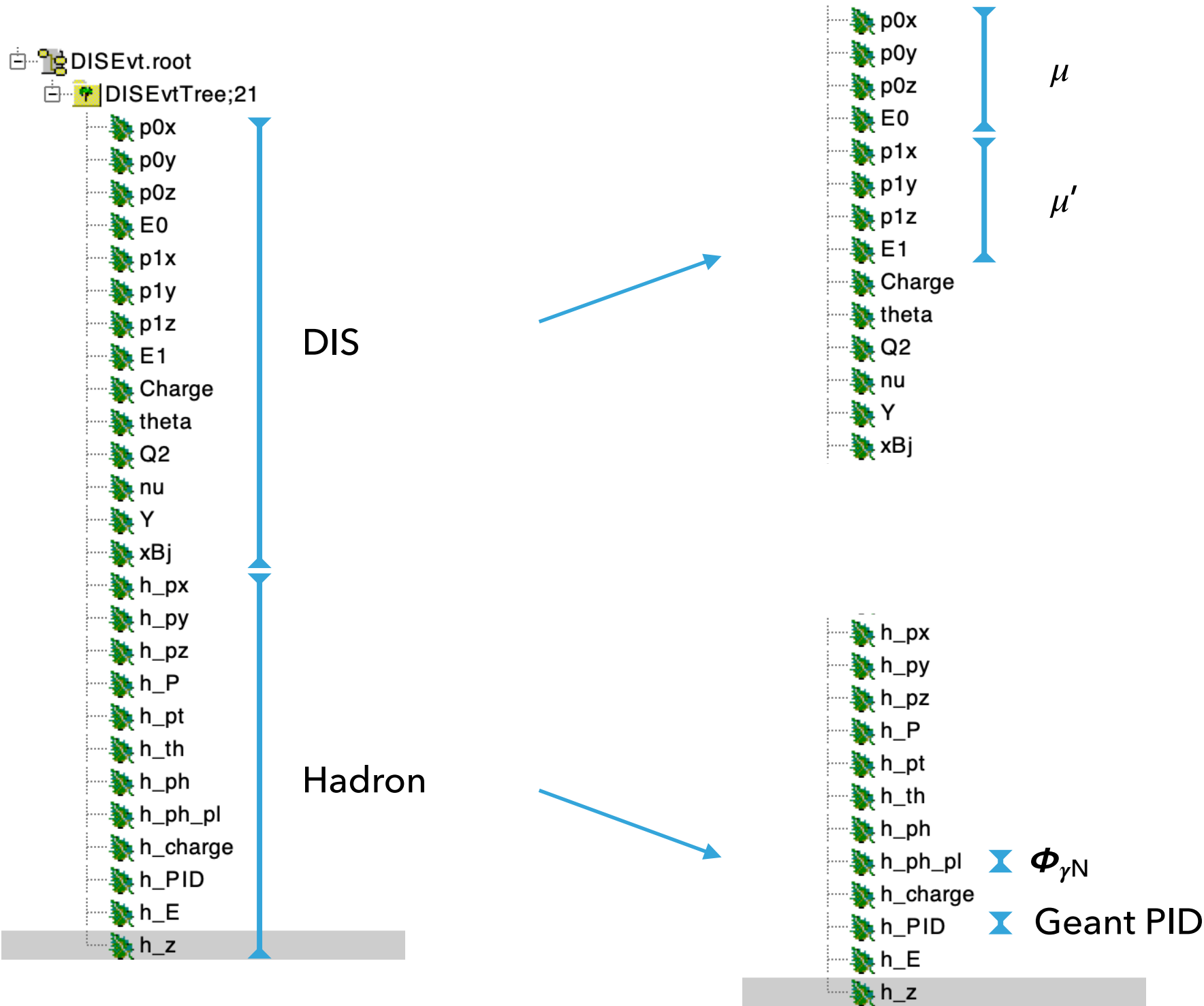


- ❖ **test.cc**, test applet.
- ❖ **xsgen.cc**, inclusive XS generation applet.
- ❖ **EvtGen.cc**, event generation applet, results stored in ROOT trees.



- ❖ **pdist.cc**, look at the kinematic distributions of events.
- ❖ **rc_calc.cc/sirc.cc/ptrc.cc**, radiative correction factors to multiplicities calculators and plotters.
- ❖ **TERAD_plot.cc**, plot TERAD inclusive corrections.





TDJANGOH available on GitHub (later today on Gitlab I guess)

<https://github.com/nipierre/TDJANGOH.git>

nipierre / TDJANGOH

Unwatch 1

Star 0

Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

A C++ Interface to DJANGOH

Edit

Manage topics

363 commits

2 branches

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find File

Clone or download

npierre

Hotfix for pt correction computation

Latest commit ab71482 on 26 Jun 2018

| | | |
|----------------|---|-------------|
| TGEANT_scripts | Added TGEANT script to retrieve DJANGOH version | last year |
| data | no message | 2 years ago |
| doc | Updated doxygen documentation. | last year |
| include | Fix. | last year |
| src | Fix | last year |

How To Install TDJANGO ?

Requirements

GCC > 4.4.7 with build-in GFORTRAN

ROOT > 5.34/32

LHAPDF 5.9.1 (in any case lower than LHAPDF 6.x.x) [Download LHAPDF 5.x.x](#)

CERNLIB (2006) [Download CERNLIB](#)

(Tested on LXPLUS and CCAGE)

Environment variables

After recovering the package, export the following variables (better in .bashrc or similar) :

```
export TDJANGO=/path/to/TDJANGO/ export LHAPDF5=/path/to/LHAPDF5/
```

Setup & Build

To setup : `make setup`

To build : `make`

TGEANT (TDJANGOH version) available on COMPASS GitLab

https://gitlab.cern.ch/compass/mc_tools/TGEANT/tree/development/djangoh

COMPASS > TGeant and MC tools > TGeant > Repository


development/dja...
development/djangoh

TGEANT / +

History

Find file

Web IDE

 **Fixed Camera Construction for 2016**
Nicolas Pierre authored 8 months ago

0ac219b6

| Name | Last commit | Last update |
|---------------------------|--|---------------|
| HEPGenPlusPlus @ c3b48267 | new DVCS 2016 alignment | 2 years ago |
| Interface | Fix. | 1 year ago |
| TGEANT | Fixed Camera Construction for 2016 | 8 months ago |
| Toolbox | Merge branch 'master' of https://gitlab.cern.ch/compass/mc_tools/TGEANT | 10 months ago |
| cmake/Modules | Merge branch 'master' of https://gitlab.cern.ch/compass/mc_tools/TGEANT | 1 year ago |
| libEvent | Test. | 1 year ago |
| libSettings | Fix filename issue with djangoh | 1 year ago |

- ❖ CMake knows where to find TDJANGO (thanks to dedicated cmake plugin)
- ❖ Implementation of one Process Class: **T4DjangohProcess.cc**.

```
void T4DjangohProcess::initDjangoh(void)
{
    if (tDjangoh != NULL)
        delete tDjangoh;

    T4SDjangoh* djangoh;
    djangoh = settingsFile->getStructManager()->getDjangoh();

    std::string cfilename = Form("%s/resources/djangoh/djangoh_input.xml",getenv("TGEANT"));

    cout << "** XML input file : " << cfilename << " **" << endl;

    tDjangoh = new TDjangoh();
    tDjangoh->ReadXMLFile(cfilename);
    tDjangoh->Initialize();
}
```

Create instance of TDJANGO

Find the XML input file within TGEANT

Initialize DJANGO

```
G4VParticleChange* T4DjangohProcess::PostStepDoIt(const G4Track& aTrack, const G4Step& aStep)
{
    aParticleChange.Initialize(aTrack);

    G4double energy = aStep.GetPostStepPoint()->GetKineticEnergy() / CLHEP::GeV;

    tDjangoh->SetBeam(energy, 0); // needs GeV

    tDjangoh->GenerateEvent();

    std::pair<G4double, G4ThreeVector> rotation = getRotation(
        aStep.GetPostStepPoint()->GetMomentumDirection());

    T4BeamData* beamData = T4EventManager::getInstance()->getBeamData();
    beamData->setDefault();

    Lujets_t* djangohParticle = tDjangoh->GetLujets();

    std::vector<G4int> lineList;
    for (G4int i = -1; i < tDjangoh->GetN()-1; i++) {
        if (djangohParticle->K[0][i] != 1)
            continue;
        G4int id = djangohParticle->K[1][i];

        if (std::abs(id) == 12 || std::abs(id) == 14 || std::abs(id) == 16) // skip neutrinos
            continue;
        else if ((std::abs(id) >= 1 && std::abs(id) <= 8) || id == 21)
            T4SMessenger::getInstance()->printMessage(T4SWarning, __LINE__, __FILE__,
                "T4DjangohProcess::PostStepDoIt: Djangoh output has a quarks/gluon (id = " + intToStr(id) + ")");
        else
            lineList.push_back(i);
    }
}
```

Set beam energy

Generate event

Take Lujets content

Loop over final state
particles

```
aParticleChange.SetNumberOfSecondaries(lineList.size());

G4ThreeVector vertexPosition = aStep.GetPostStepPoint()->GetPosition();
G4double vertexTime = aStep.GetPostStepPoint()->GetGlobalTime();

//Workaround in order to recover infos about incoming muon and nucleus

for (G4int i = 0; i < 2; i++)
{
    G4int id;
    G4ThreeVector helper;
    if(!i)
    {
        id = tDjangoh->GetIDPHEP(1);
        helper = G4ThreeVector(0,0,energy);
    }
    else
    {
        id = tDjangoh->GetNucleusID();
        helper = G4ThreeVector(0,0,0);
    }

    if (rotation.first != 0)
        helper.rotate(rotation.first, rotation.second);

    T4BeamParticle particle;

    particle.k[0] = 21;
    particle.k[1] = id;
    particle.k[2] = 0;
    particle.k[3] = 0;
    particle.k[4] = 0;
```

Workaround to have
incoming particles in
the list


```
particle.p[0] = helper.x();
particle.p[1] = helper.y();
particle.p[2] = helper.z();
if(!i)
{
    particle.p[3] = energy;
    particle.p[4] = tDjangoh->GetPHEP(5,1);
}
else
{
    particle.p[3] = 0;
    particle.p[4] = 0.938272;
}

beamData->beamParticles.push_back(particle);
}
```

```
for (G4int i = -1; i < tDjangoh->GetN()-1; i++)
{
    G4int id = djangohParticle->K[1][i];
    G4ThreeVector helper = G4ThreeVector(djangohParticle->P[0][i],
        djangohParticle->P[1][i], djangohParticle->P[2][i]);

    if (rotation.first != 0)
        helper.rotate(rotation.first, rotation.second);

    T4BeamParticle particle;
    if (std::find(lineList.begin(), lineList.end(), i) != lineList.end()) {

        G4ThreeVector momentumDirection = helper.unit();
```

Loop over final state
particles

Rotation

```
G4double kinEnergy = djangohParticle->P[3][i] * CLHEP::GeV; //pythia convention(?!): p[3] is set to

if (kinEnergy > djangohParticle->P[4][i] * CLHEP::GeV) {
    G4DynamicParticle* aParticle = new G4DynamicParticle(getParticleByID(id),
        momentumDirection, kinEnergy);

    G4ThreeVector vertexHelper = G4ThreeVector(

        djangohParticle->V[0][i] * CLHEP::mm,
        djangohParticle->V[1][i] * CLHEP::mm,
        djangohParticle->V[2][i] * CLHEP::mm);

    if (rotation.first != 0)
        vertexHelper.rotate(rotation.first, rotation.second);

    G4Track* aSecondaryTrack = new G4Track(aParticle,
        vertexTime,
        vertexPosition);
    aSecondaryTrack->SetTouchableHandle(aTrack.GetTouchableHandle());
    aSecondaryTrack->SetParentID(aTrack.GetTrackID());
    aParticleChange.AddSecondary(aSecondaryTrack);
}

particle.k[0] = djangohParticle->K[0][i];
particle.k[1] = id;
particle.k[2] = djangohParticle->K[2][i];
particle.k[3] = djangohParticle->K[3][i];
particle.k[4] = djangohParticle->K[4][i];

particle.p[0] = helper.x();
particle.p[1] = helper.y();
particle.p[2] = helper.z();
```

Pass infos to TGEANT
about these particles

```
particle.p[0] = helper.x();
particle.p[1] = helper.y();
particle.p[2] = helper.z();
particle.p[3] = djangohParticle->P[3][i];
particle.p[4] = djangohParticle->P[4][i];

beamData->beamParticles.push_back(particle);
}

beamData->nBeamParticle = beamData->beamParticles.size();
beamData->generator = eventGenId;
setThreeVector<double>(beamData->vertexPosition, vertexPosition.x(),
    vertexPosition.y(), vertexPosition.z());
beamData->vertexTime = vertexTime;

beamData->x_bj = tDjangoh->GetX();

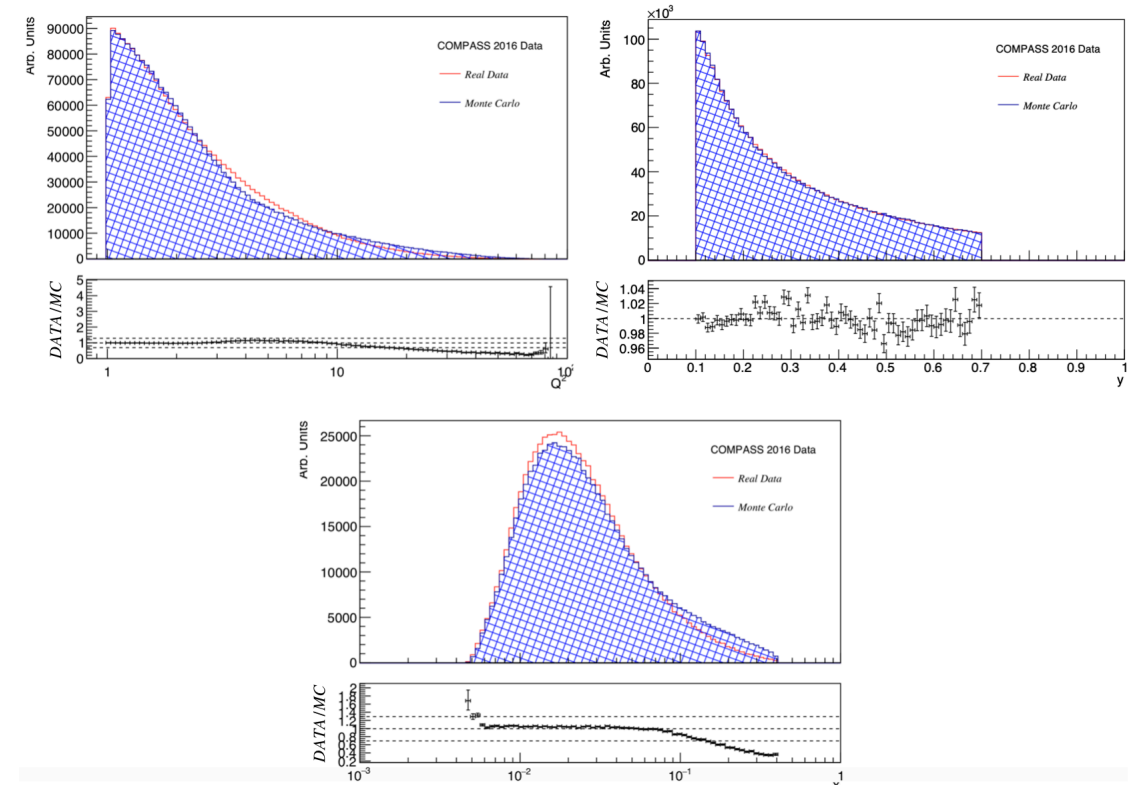
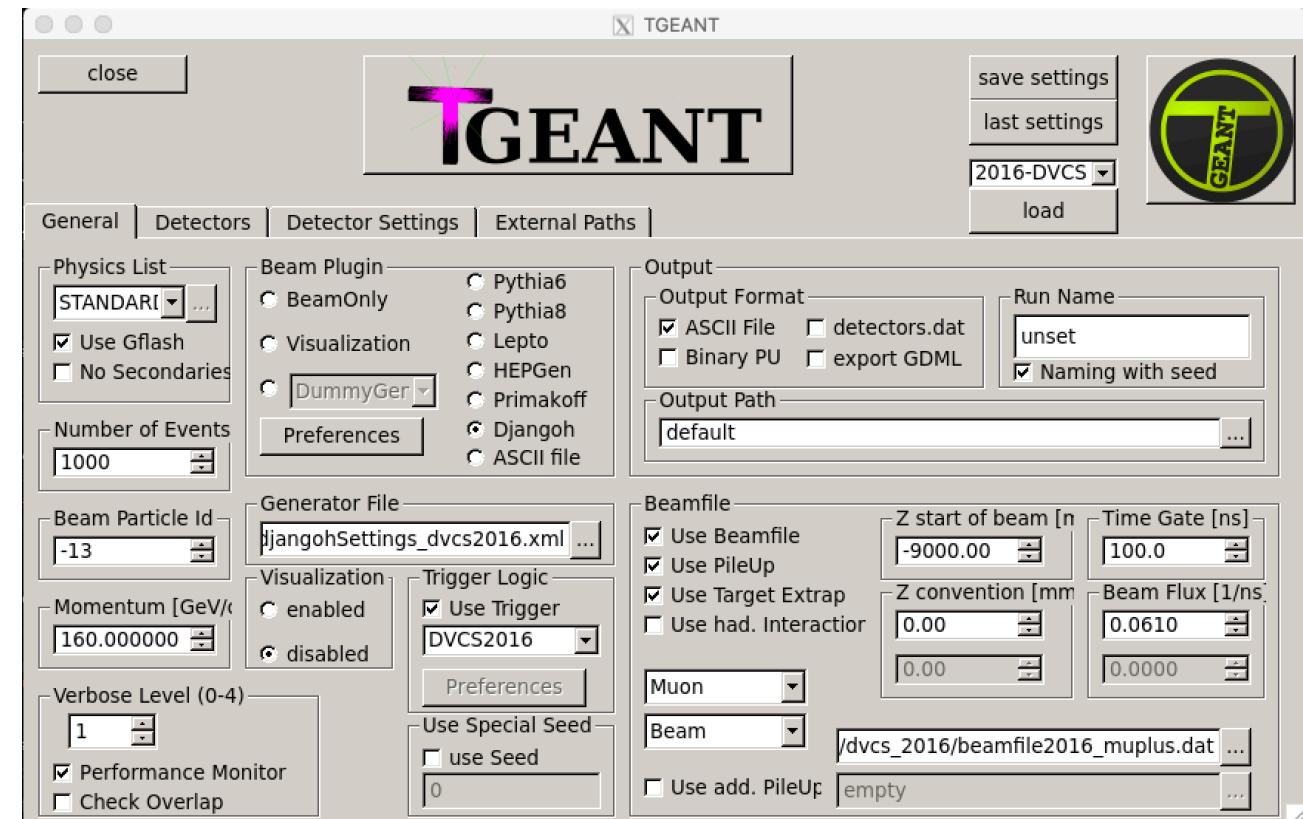
aParticleChange.ProposeTrackStatus(fStopAndKill);
targetTracking->processCalled();

T4SMessenger::getInstance()->printMessage(T4SVerboseMore, __LINE__,
    __FILE__, "T4DjangohProcess::PostStepDoIt: T4DjangohProcess was executed.");

return G4VDiscreteProcess::PostStepDoIt(aTrack, aStep);
}
```

Save infos

- ❖ Be sure to have the right configuration of DJANGO:
 - ❖ Look at `$TGEANT/resources/djangoh/djangoh_input.xml` and see if the configuration is what you expect
- ❖ Use DJANGO as event generator
 - ❖ Just check DJANGO in the interface for example (or modify the setting file from TGEANT in the according entry)
- ❖ Then generate your MC !
- ❖ Nevertheless, known issue:
 - ❖ Cannot use the same TGEANT to do DJANGO and PYTHIA: conflict of PYTHIA versions with the one from TGEANT superseding the one from DJANGO and making it crash. Found no way to force the right version to be used (but haven't put too much thought in it either).



How To Install TGEANT (TDJANGO version) ?

- ❖ **Two ways:**
 - ❖ By yourself if you're brave enough !
 - ❖ Using the little scripts from TDJANGO (perhaps they'll be moved elsewhere in the future IDK).
- ❖ **If you followed successfully the installation of TDJANGO, TGEANT will be able to find it with no problems.**

