tutorial session

UCLouvain Delphes team

Université catholique de Louvain
Louvain-la-Neuve, Belgium

Center for Particle Physics and
Phenomenology (CP3)

January 15, 2013

Introduction

- We are presenting the new modular version of Delphes
- Website:

  https://cp3.irmp.ucl.ac.be/projects/delphes
- Documentation for the new modular version:

  https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook

Installing Delphes from Source

- To successfully build Delphes the following prerequisite packages should be installed:
  - ROOT Data Analysis Framework
  - Tcl scripting language

- Setup ROOT environment variables

- Get the code:

```
wget --no-check-certificate https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WikiStart/Delphes-3.0.0.tar.gz

tar -zxf Delphes-3.0.0.tar.gz
```

- Compile:

```
cd Delphes-3.0.0

make -j 4
```

- Run:

```
./DelpheHepMC examples/delphes_card_CMS.tcl output.root input.hepmc
```

Running Delphes

- Command line parameters:

```
DelphesHepMC config_file output_file [input_file(s)]

 config_file - configuration file in Tcl format

 output_file - output file in ROOT format,

 input_file(s) - input file(s) in HepMC format,

 with no input_file, or when input_file is -, read standard input.
```

Running Delphes, cont.

- Running Delphes with HepMC input files:

```
./DelphesHepMC examples/delphes_card_CMS.tcl output.root input.hepmc
```

- Running Delphes with STDHEP (XDR) input files:

```
./DelphesSTDHEP examples/delphes_card_CMS.tcl delphes_output.root input.hep
```

- Running Delphes with LHEF input files:

```
./DelphesLHEF examples/delphes_card_CMS.tcl delphes_output.root input.lhef
```

- Running Delphes with files stored in CASTOR:

```
rfcat /castor/cern.ch/user/d/demine/test.hepmc.gz | gunzip |
./DelphesHepMC examples/delphes_card_CMS.tcl delphes_output.root
```

- Running Delphes with files accessible via HTTP:

```
curl –s http://cp3.irmp.ucl.ac.be/~demin/test.hepmc.gz | gunzip |
./DelphesHepMC examples/delphes_card_CMS.tcl delphes_output.root
```

Hands-on session

- Hands-on session outlook:
  - Example 1: creating and analyzing ROOT tree
  - Example 2: modifying configuration file
  - Example 3: adding new module
- Step-by-step instructions:

  http://cp3.irmp.ucl.ac.be/~demin/DelphesTutorial.txt

UCL
Université
catholique
de Louvain

Example 1: creating and analyzing ROOT tree

ROOT tree: structure

- ROOT tree is a set of branches
- Branch contains a collection of analysis objects for every event:
  - information is stored in TclonesArray, which enables efficient storage an retrieval
  - same C++ classes (Jet, Muon etc.) used for creating the tree and for analyzing the stored data
  - different quantities stored in the tree can be linked by pointers
- ROOT tree description:

  http://cp3.irmp.ucl.ac.be/~demin/RootTreeDescription.html

ROOT tree: ExRootTreeReader

- ExRootTreeReader – class simplifying access to ROOT tree data
  - ExRootTreeReader(TTree *) – constructor takes ROOT tree as parameter
  - Long64_t GetEntries() – returns total number of events stored on the tree
  - TClonesArray *UseBranch(branchName, className) – returns pointer to collection of branch elements and specifies branches needed for analysis
  - Bool_t ReadEntry(entry) – loads collections of branch elements with data from specified event

UCL Université catholique de Louvain

Example 2: modifying configuration file

Modules

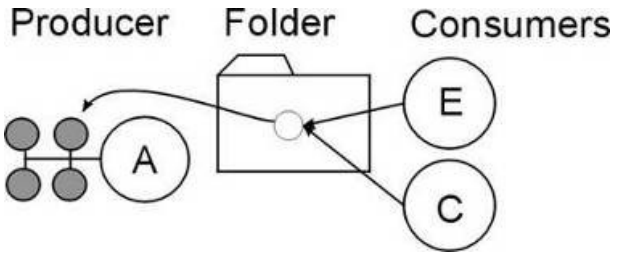
- Module system is inspired by the 'Folders and Tasks' chapter from the ROOT Users Guide
  http://root.cern.ch/download/doc/ROOTUsersGuideHTML/ch10.html

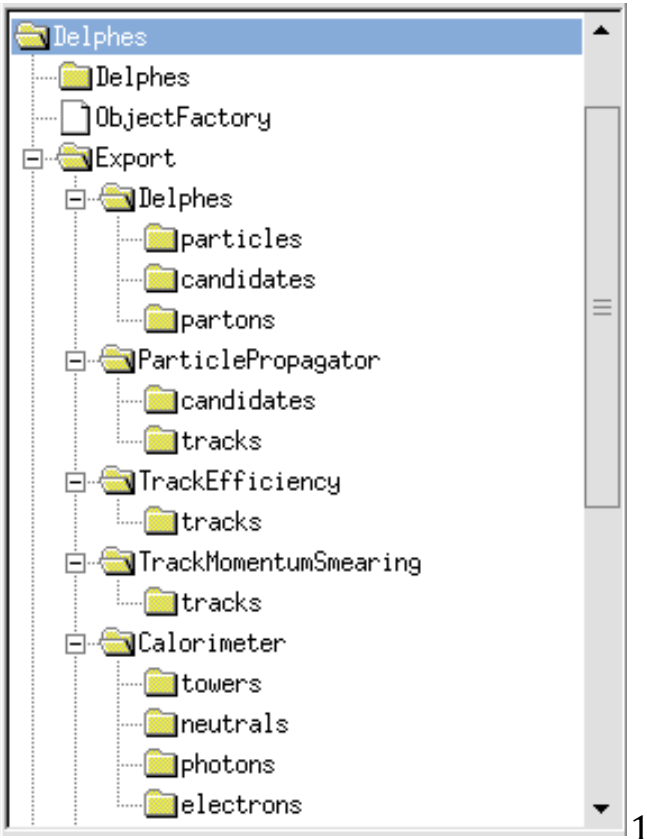
- It's based on the TTask and TFolder classes
  http://root.cern.ch/root/html/TTask.html
  http://root.cern.ch/root/html/TFolder.html

- All modules consume and produce TObjArrays of Candidates

- Every module has a corresponding TFolder containing TObjArrays produced by this module

- Any module can access TObjArrays produced by other modules using ImportArray method:
  ImportArray("ModuleName/arrayName")





Configuration file

- Delphes' configuration file is based on Tcl scripting language
- Basic commands:
  - set value of a scalar parameter:

```
set ParamName value
```

  - append a list of values to a vector (list) parameter

```
add ParamName value1 value2 value3 ...
```

  - activate a module:

```
module ModuleClass ModuleName ModuleConfigurationBody
```

  - define order of execution of various modules:

```
set ExecutionPath ListOfModuleNames

set ExecutionPath {
  ParticlePropagator
  TrackEfficiency
  TrackMomentumSmearing
  ...
  TreeWriter
}
```

Module configuration example

```
module Efficiency ElectronEfficiency {
  set InputArray ElectronEnergySmearing/electrons

  set OutputArray electrons

  # add EfficiencyFormula {abs(PDG code)} {efficiency formula as a function of eta and pt}

  # default efficiency formula
  add EfficiencyFormula {0} {0.0}

  # efficiency formula for electrons
  add EfficiencyFormula {11} {                                    (pt <= 5.0) * (0.000) + \
                                          (abs(eta) <= 1.5) * (pt > 5.0)  * (0.950) + \
                              (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 5.0)  * (0.850) + \
                              (abs(eta) > 2.5)                                  * (0.000)}
}
```

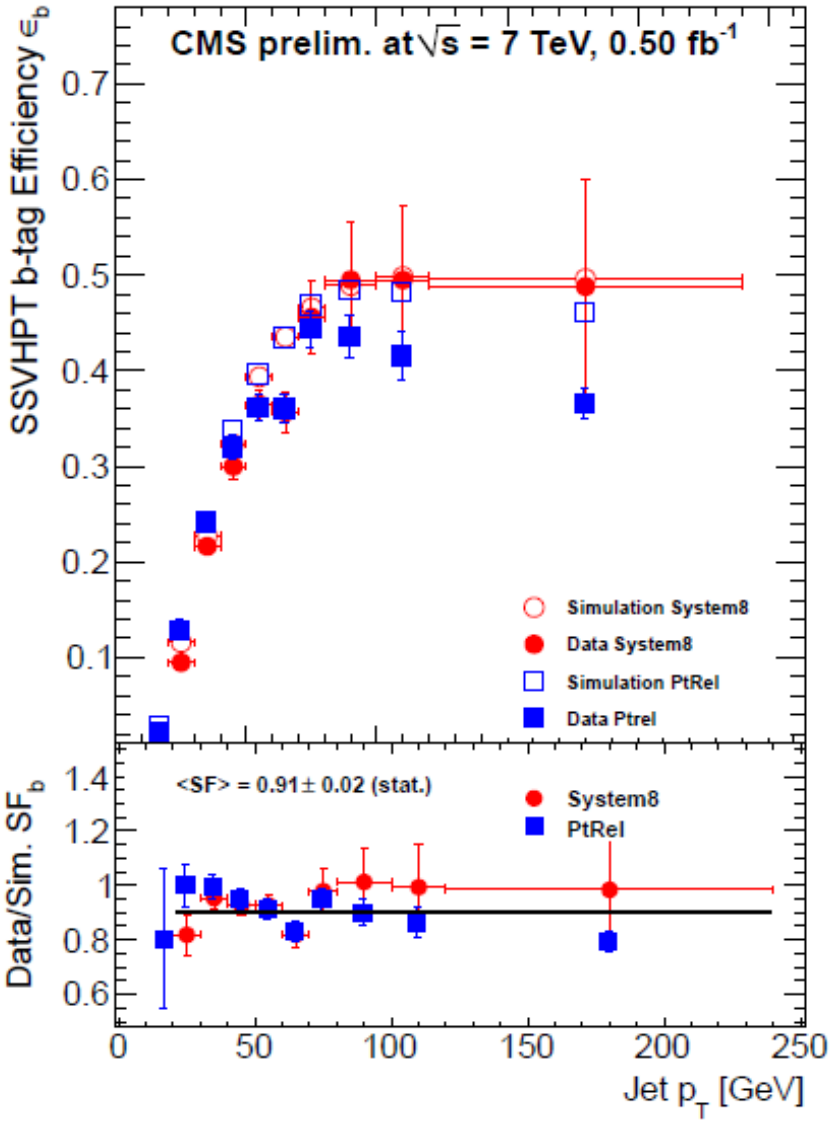
All efficiency, resolution, etc formulas can be configured as function of E, p_T, η, φ

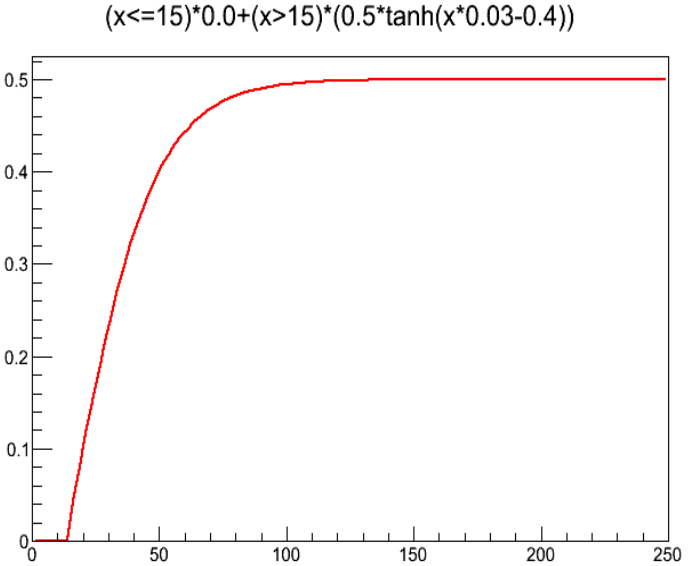
using ROOT's TFormula syntax:

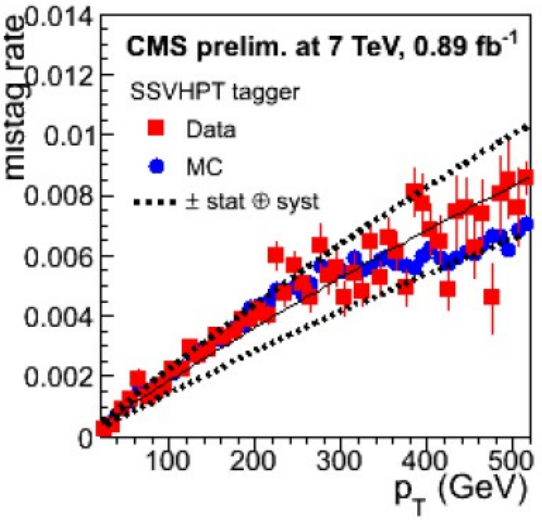
http://root.cern.ch/root/html/TFormula.html

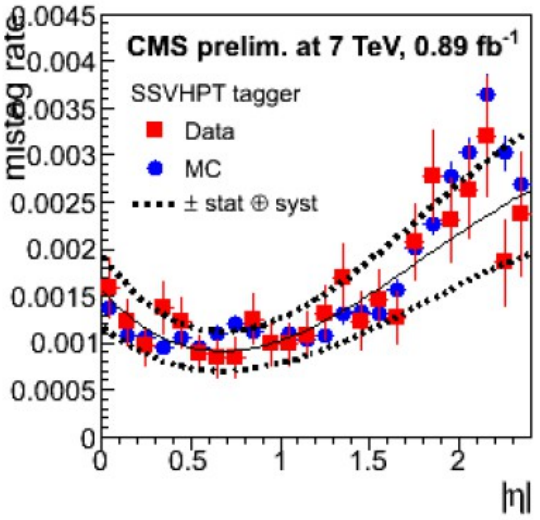
Let's modify the configuration file

- We can try to tune b-tagging performance according to the CMS public plots https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhysicsResultsBTV









Tuned b-tagging performance

- We will replace the default b-tagging efficiency formulas with the the new ones:

```
# default efficiency formula (misidentification rate)
add EfficiencyFormula {0} {0.001}

# efficiency formula for c-jets (misidentification rate)
add EfficiencyFormula {4} {                                    (pt <= 15.0) * (0.000) + \
                                          (abs(eta) <= 1.2) * (pt > 15.0) * (0.2*tanh(pt*0.03 - 0.4)) + \
                          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.1*tanh(pt*0.03 - 0.4)) + \
                          (abs(eta) > 2.5)                                * (0.000)}

# efficiency formula for b-jets
add EfficiencyFormula {5} {                                    (pt <= 15.0) * (0.000) + \
                                          (abs(eta) <= 1.2) * (pt > 15.0) * (0.5*tanh(pt*0.03 - 0.4)) + \
                          (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.4*tanh(pt*0.03 - 0.4)) + \
                          (abs(eta) > 2.5)                                * (0.000)}
```

Example 3: adding new module

Adding new module

- Steps needed for adding new module:
  - copy template/example module to the new one:

```
cp modules/ExampleModule.h modules/NewModule.h
cp modules/ExampleModule.cc modules/NewModule.cc
```

  - edit new module:

```
vi modules/NewModule.h
vi modules/NewModule.cc
```

  - add new module to modules/ModulesLinkDef.h:

```
#include "modules/NewModule.h"
...
#pragma link C++ class NewModule+;
```

  - regenerate Makefile:

```
./configure
```

  - rebuild Delphes:

```
make -j 4
```